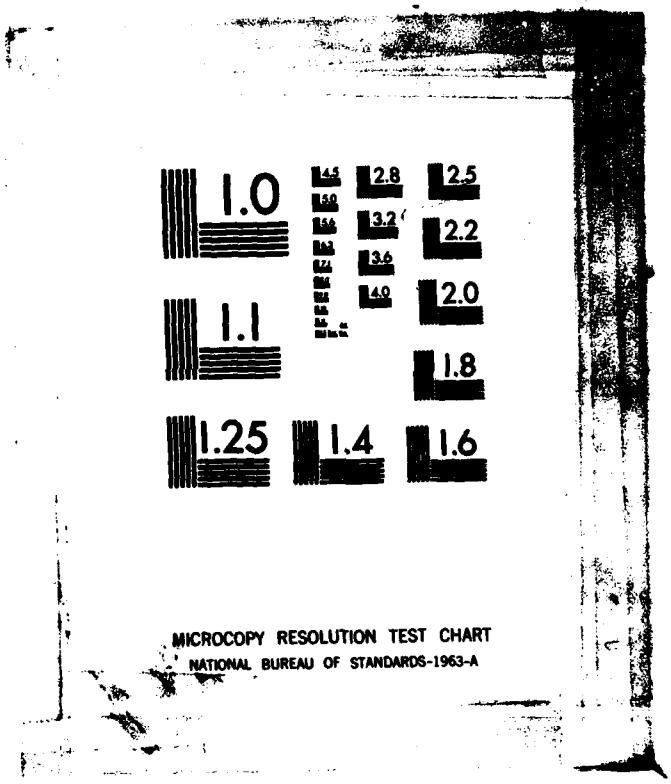
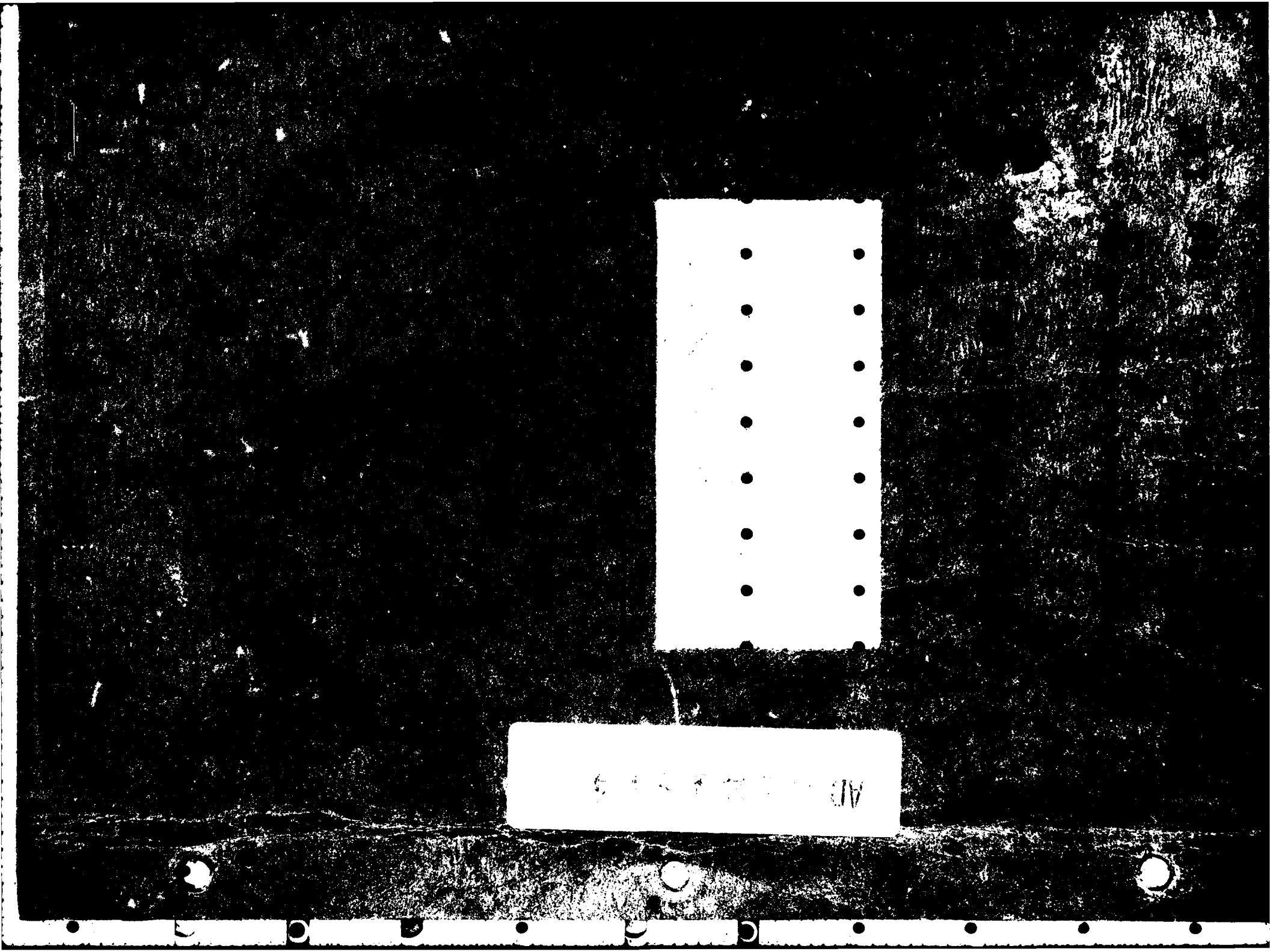
AD-A124 814 PERFORMANCE OF SEQUENTIALLY DECODED LONG CONSTRAINT LENGTH CODES FOR A SA.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J A FRAZIER 1/3

UNCLASSIFIED DEC 82 AFIT/GE/EE/82D-32 F/G 17/2.1 NL


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



AFIT/GE/EE/82D-32

PERFORMANCE OF SEQUENTIALLY DECODED LONG CONSTRAINT LENGTH CODES FOR A SATELLITE COMMUNICATIONS LINK IN A SCINTILLATED CHANNEL

THESIS

AFIT/GE/EE/82D-32 James A. Frazier, Jr.
Capt USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/82D-32

PERFORMANCE OF SEQUENTIALLY DECODED LONG CONSTRAINT LENGTH CODES FOR A SATELLITE COMMUNICATIONS LINK IN A SCINTILLATED CHANNEL

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
James A. Frazier, Jr.
Captain USAF
Graduate Electrical Engineering
December 1982

Preface

The purpose of this thesis was to evaluate the performance of sequentially decoded long constraint length codes for a satellite communications link operating through a scintillated channel. I feel the resultant comparisons made between this convolutionally encoded system and the Viterbi decoded implementation will provided insight for channel coding decisions in the development of future tactical/strategic military satellite communication systems.

I would like to express my deepest gratitude to my advisor, Major Kenneth Castor for his dedicated and enthusiastic technical direction throughout the entire project. In addition, I would like to thank the Flight Dynamics Laboratory at Wright-Patterson AFB, Ohio for allowing me to use their computer facilities.

Special thanks is due to Captain Gary Krajci, Satellite and C^3 Branch, Air Force Weapons Laboratory at Kirtland AFB, New Mexico for providing this thesis topic, and AFWL contractor, Mission Research Corporation, Santa Barbara, California for their technical support.

Finally, I want to express my sincere appreciation in dedication of this thesis to my wife, Kathy and daughter, Jeanine for their love, patience and support during this effort. Their continuous encouragement and understanding made this thesis a special success.

James A. Frazier, Jr.

Contents

List of Figures

List of Tables

Abstract

Convolutional codes with long constraint lengths, on the order of 25 to 30, can be decoded using the Fano sequential decoding algorithm. While some increase in coding gain over short codes may be achieved for small probabilities of bit error, sequential decoding is not optimum and does not perform as well as the Viterbi decoding algorithm. However, because of power limitations on satellite downlinks, the possibility of achieving some additional gain with long constraint length codes for the scintillated signal channel is receiving increasing interest.

This report presents an analysis of the performance and implementation trade-offs between short codes with Viterbi decoding and long codes with sequential decoding in an additive white Gaussian noise channel and a scintillation channel created by a high-altitude nuclear detonation. Performance is provided for and EHF, 2400 bps, DBPSK modulated satellite communications link in terms of bit error rates and decoded bit processing delays.

Performance comparisons between the two convolutionally encoded systems show Viterbi decoding outperforms the sequentially decoded link for scintillated signal fading channels at bit error probabilities greater than 10^{-5}. The Viterbi decoded link, however, loses its advantage over the sequentially decoded systems at bit error rates less than 10^{-5} in slow fading channel conditions.

I. Introduction

Background

Future military satellite communication systems serving nuclear capable forces for command, control and communication purposes require the highest degree of electromagnetic survivability. Electromagnetic survivability includes protection against nuclear burst induced signal scintillation effects which can degrade the performance of satellite communication links. These satellite/C^3 systems typically require low to medium data rates and strongly emphasize the use of extremely high frequencies (EHF) links.

Nuclear explosions produce a number of signal propagation disturbances that seriously threaten satellite link performance. Resulting signal disturbances include absorption, noise, dispersion, Doppler shift, time delay, refraction, angular scattering and scintillation. The spatial extent and duration of propagation disturbances depend primarily on the nuclear burst parameters (height of burst, number and location of burst, weapon yield) as well as on signal carrier frequency and propagation path geometry.

The significance of propagation effects on system performance depends strongly on link design (carrier frequency, data rate, type of modulation, coding and tracking loop configuration) as well as on power margin and system mission requirements. Some degradation of system performance,

relative to that in normal undisturbed environments, is generally inevitable in a nuclear-disturbed propagation environment. However, proper attention to the nuclear propagation threat and careful application of practical engineering techniques can provide hardened link designs that achieve acceptable levels of survivability in nuclear environments.

Convolutional error correction codes are becoming widely used to improve the efficiency of satellite links and other radio communication channels. For example, maximum likelihood decoding of such codes using the Viterbi algorithm can provide a significant coding gain.

Greater coding gains are, in principle, achievable using codes with longer constraint lengths. In terms of performance, the Viterbi maximum-likelihood algorithm is the optimum method of decoding such codes. However, the computational complexity of Viterbi decoding increases exponentially with the constraint length. Thus Viterbi decoding becomes impractical with long constraint length codes, on the order of 10 or more.

Codes with quite long constraint lengths, on the order of 25 to 30, can be decoded using a form of the sequential decoding algorithm. While some increase in coding gain over short codes may be achieved, the sequential decoding algorithm is not optimum and does not perform as well as the Viterbi algorithm. However, because of power limitations on satellite downlinks, the possibility of achieving some additional

gain with long constraint length codes is receiving increasing interest.

The need for increased coding gain is particularly acute when the link must operate in the presence of severe scintillation or fading. Signal scintillation is frequently encountered at UHF in the natural ionosphere, especially in the equatorial and auroral regions. Instances of naturally occuring signal scintillation have also been observed well in the gigahertz frequency band. In a nuclear environment, signal scintillation can be encountered at all frequencies used by satellite systems. There is, therefore, an increasing need for techniques that improve link efficiency in both undisturbed and disturbed propagation channels. This need has led to renewed interest in long constraint length codes which in turn require the use of sequential decoding.

Problem Statement

The effort is directed toward an analysis of the performance and implementation trade-offs between a short convolutional code with Viterbi decoding and long convolutional codes with sequential decoding in additive white Gaussian noise (AWGN) and signal scintillation fading conditions. These two convolutional forward error correctional coding techniques are to be implemented for a 2400 bps, differentially coherent binary phase-shift-keyed (DBPSK) satellite communication data link. Performance assessments are to be based upon digital

link simulations using the Air Force Weapons Laboratory (AFWL) FSK-PSK Link Performance Code. The emphasis in the final product of this work is to be a comparative performance analysis for the Viterbi and sequential decoded satellite links stressed to nuclear burst induced signal disturbances. System performance implications based upon implementation trade-offs are to be addressed for the scintillated channel as well.

Limitations

The performance analysis is given in terms of bit error probability versus the received energy-to-noise density ratio. Lower bound on bit error rate is 10^{-5}. Implementation trade-offs are evaluated in terms of decoder bit processing delays.

Quantitative analyses of the performance of the Viterbi decoded link versus the sequential decoded links are given for the AWGN and scintillated signal channels. The link simulations for the scintillated channel are done for signal fading periods or decorrelation times (τ_0) equal to 0.016 sec , 0.05 sec and 0.16 sec , corresponding to moderately fast, moderately slow and slow fading channels respectively.

The Viterbi decoded link uses a rate 1/2, constraint length 7 convolutional code and the long constraint length codes used in the sequential decoded links are rate 1/2,

constraint length 25; and rate 1/2, constraint length 30. Satellite link parameters include a 20 GHz carrier frequency, 2400 bps data rate, differential encoding, differentially coherent binary phase-shift-key modulation/demodulation, synchronous interleaving and deinterleaving of data, as well as, the implementation of automatic gain control and automatic frequency control of the received signal.

Approach and Presentation

The approach for generating the performance and implementation trade-offs between the Viterbi and sequential decoding schemes involved detailed link simulations using the AFWL FSK-PSK Link Performance Code. This code produces the nuclear channel characteristics which create a scintillated signal structure and simulates the processes of modulation/demodulation, synchronous interleaving/deinterleaving and forward error-correction channel coding of digital data. The AFWL Code also measures satellite link performance in terms of bit error rates.

The FSK-PSK Link Performance Code already had the capability to convolutionally encode a desired short constraint length code ($K \leq 8$) and perform Viterbi decoding. Therefore, the first major task involved designing a FORTRAN subroutine to emulate a sequential decoder.

This effort required the selection of "good" rate 1/2 long constraint length codes and, more importantly, a sequential decoding algorithm. The sequential decoding algorithm

had to be efficient and compatible with the AFWL Code. After the algorithm was selected, the sequential decoder subroutine was designed, implemented and tested. These last three tasks were the most difficult and time consuming parts of the project.

The second major task consisted of performing the actual link simulations to obtain performance results for the convolutional encoded systems in AWGN and scintillation conditions. Lastly, the performance results were analyzed and implementation trade-offs for the Viterbi and sequential decoded systems developed.

We begin in Chapter II with a general discussion of nuclear induced signal scintillation effects on satellite communication links. Then in Chapter III, we learn about the major components of the Link Performance Code and how DBPSK link simulations are done.

Next, in Chapter IV we move to convolutional codes and, in particular, the specific codes implemented in this investigation. In Chapter V, the entire sequential decoder design and implementation effort discussed above is presented. Then a brief description of Viterbi decoding is given in Chapter VI.

Before presenting the results of this project, synchronous interleaving is discussed in Chapter VII. This is then followed by results presented in Chapter VIII; and conclusions and recommendations in Chapter IX.

II. Signal Scintillation Effects on the Digital Satellite Communications Link

High-Altitude Nuclear Environment

High-altitude nuclear explosions produce greatly increased levels of ionization which cause a variety of signal propagation disturbances. Such propagation disturbances include absorption, phase shift, time delay, dispersion, polarization, rotation, refraction, multipath and increased noise levels (Ref 7:21). If, in addition, the artificially enhanced ionospheric electron concentration becomes structured or striated, then the propagating signal is also subject to fluctuation or scintillations in phase and amplitude.

Nuclear detonations produce intense ionized regions due to deposition of much of the radiated energy in the ionosphere. The spatial extent, location and duration of the ionization are sensitive functions of detonation altitude and also depend on weapon yield and other characteristics. At high detonation altitudes, above 100 km, the weapon energy is deposited in a large volume, producing long-lasting ionization in widespread regions of the ionosphere. Such high-altitude detonations cause the plasma to become structured, or striated, throughout large regions that extend to extremely high altitudes along the earth's magnetic field. This small structure which resembles filaments of high electron content is primarily created by an instability generated

by a high-altitude neutral wind blowing in the direction of decreasing electron concentration. Therefore, striations are irregular ionization structures and refer to fluctuations in electron density on a scale smaller than the scale of the initial radiation deposition by a high-altitude nuclear detonation.

Striations extend to extremely high altitudes on the order of 10,000 to 30,000 km. The horizontal extent of the plasma can be several hundred kilometers at the lower altitudes and considerably larger at the higher altitudes due to the divergence of the geomagnetic field lines. Nuclear striations can have a significant impact on electromagnetic signal propagation transversing the large spatial regions affected by the burst. In addition, fully developed striations will exist within a few tens of minutes, and once formed, will persist for time periods of several hours.

Signal Scintillation Characteristics

In the nuclear disturbed ionosphere, random fluctuations of electron density (striations) cause radio waves to be scattered, producing random variations in received signal amplitude and phase. Such signal variations are called scintillations or fading. If all the frequency components of the received signal vary essentially identically with time, the propagation channel is referred to as time selective or flat fading. When the scintillations exhibit statistical

decorrelation at different frequencies within the signal bandwidth, the channel is referred to as frequency selective. Because of the relatively narrow bandwidth of the 2400 bps, DBPSK link considered in this thesis, the time selective fading response of the system was the only major signal scintillation condition to be analyzed.

Figure 1 illustrates the geometrical nature of the problem. Propagation of satellite signals through a large striated region presents the problem of radio wave propagation through a thick medium composed of random fluctuations in the index of refraction. This problem can best be demonstrated by considering an unmodulated carrier wave traversing the striated region. The wave first suffers random phase perturbations due to variations in the phase velocity within the medium. As the wave propagates further, diffractive effects introduce fluctuations in amplitude as well as phase. These time-varying amplitude and phase scintillations represent an undesired complex modulation of the carrier.

Figure 2 represents a typical scintillated signal structure. The amplitude scintillations seen over this small portion of time are shown at the top, with the corresponding phase scintillations at the bottom. Note that many (if not all) of the amplitude fading spikes in the top portion of the figure correspond in time with the phase spikes or glitches at the bottom. It is the combined effect of the amplitude and phase scintillation which causes receiver degradation

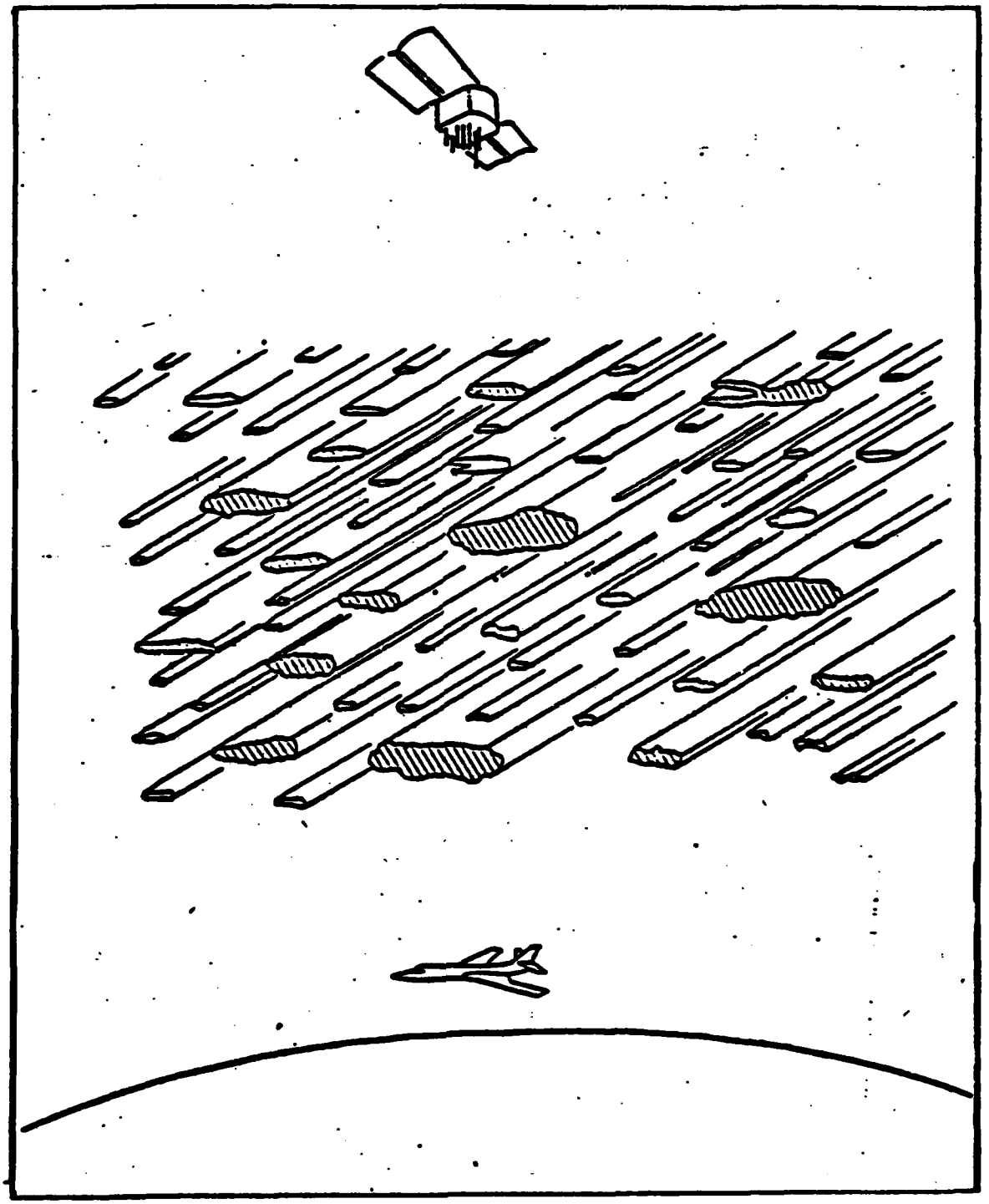
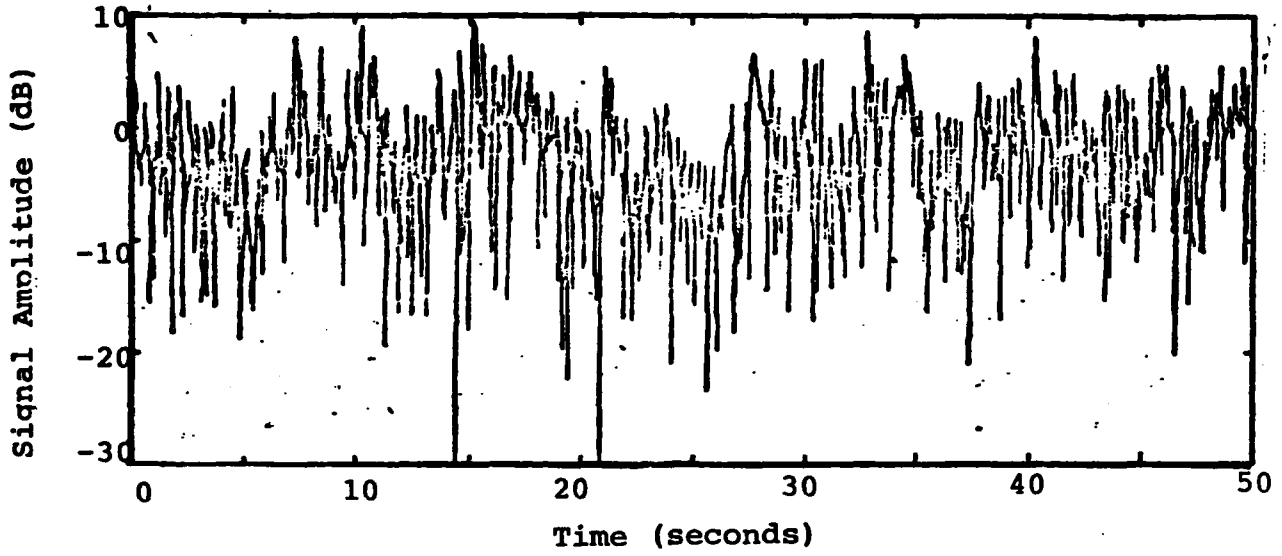


Figure 1. Satellite Communications Through a Striated Environment.



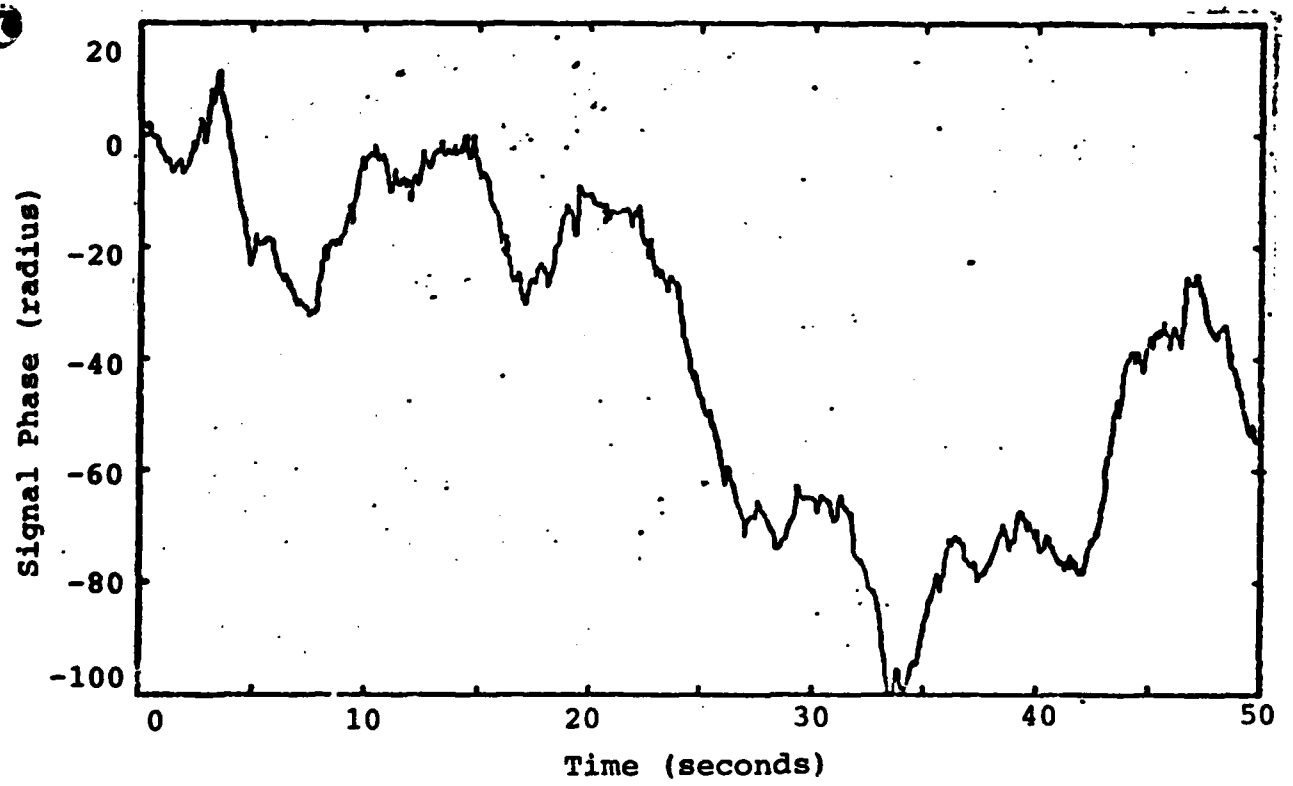


Figure 2. Example of Moderately Fast Fading Received Signal Structure (Ref 7:156).

beyond that which occurs in the normal undisturbed additive white Gaussian noise (AWGN) environment.

Nuclear environment and signal propagation studies have characterized scintillated signal amplitude fluctuations as Rayleigh distributed and the combined amplitude and phase fluctuations are approximated by an f^{-4} signal power spectrum (Ref 7:34, 122-126) and (Ref 28:2). A detailed discussion of scintillated signal structure specification is provided by Wittwer (Ref 28:1-4).

The most important signal parameter affecting receiver performance for the time selective fading channel is the scintillation fading rate. This is measured by the signal decorrelation time, τ_o. Values of τ_o are likely to be anywhere in the range from around a millisecond to about 10 seconds. Large values of τ_o represent slow fading and smaller values of τ_o are applicable to rapid or fast fading. The structure in Figure 2 has a moderately fast fading rate ($\tau_o \cong 0.5$ sec) .

More precisely, τ_o is the signal decorrelation time defined to be the e^{-1} point on the time autocorrelation function of the complex electric field at the receiver (Ref 7: 137). The parameter τ_o is related to the signal correlation distance l_o and an effective velocity V_{eff} by

$$\tau_o = \frac{l_o}{V_{eff}} \text{ sec} \qquad (1)$$

The signal correlation distance l_o is similarly defined as the e^{-1} point on the spatial autocorrelation function of the complex signal in the receiver plane. That is, it is the distance from a point on the receiver plane in which the complex signal falls to a value of e^{-1} times its nominal received signal level.

The effective velocity in Eq (1) is a weighted average of the component of the relative velocity between the propagation path and striated medium in a direction normal to the path and normal to the geomagnetic field-aligned striation axes. Thus this effective velocity is a function of many system and environmental parameters. These include satellite and airborne terminal velocities, striation drift velocities, orientation of the geomagnetic field and propagation path geometry (Ref 7:41).

An additional way to define τ_o is first to define the complex modulation envelope of the received signal that was imposed by the propagation medium as $E(t) = E_o + E_s(t)$, where E_o is the mean signal component and $E_s(t)$ is the random scintillating component. Then $R_s(\tau) = \overline{E_s(t)\ E_s^*\ (t+\tau)}$ is the autocorrelation function of the random component, and τ_o is defined such that $R_s(\tau_o) = e^{-1}R_s(0)$ (Ref 6:37).

Signal scintillation is a widespread and long-lasting propagation disturbance in a high-altitude nuclear environment. A single detonation can produce intense UHF scintillation throughout regions that extend over CONUS-sized areas.

The spatial extent and duration of significant satellite link disturbances caused by high-altitude nuclear burst decrease with increasing frequency (Ref 12:2). Figure 3 compares the predicted extent of intense scintillation at UHF, L-Band (1.5 GHz), S-Band (2.25 GHz), X-Band (7.5 GHz) and EHF (20 to 45 GHz). At UHF signal scintillation effects are likely to persist for several hours.

In general, signal scintillation will decrease in severity with increasing frequency, but remain potentially significant even at EHF. A 20 GHz satellite communications link like the one analyzed in this report could be subject to a scintillation region that may extend a few hundred kilometers and persist up to an hour after a high-altitude nuclear burst. In addition, the range of τ_0 for a 20 GHz link is: $0.013 \text{ s} \leq \tau_0 \leq 0.90 \text{ s}$ (Ref 12:8). The range and time extent of τ_0 for a carrier frequency is shown in Figure 4.

Signal Scintillation Effects on the Medium Data Rate Link

The purpose of this section is to present an overview of the dominant effects of time selective signal scintillation on a medium data rate (2400 bps) satellite communication link. Emphasis here is on the uncoded link but the interest in the coded link as a mitigation technique will be discussed as well.

Medium-data-rate links are severely degraded in rapid fading (small τ_0) when the signal scintillation rate exceeds

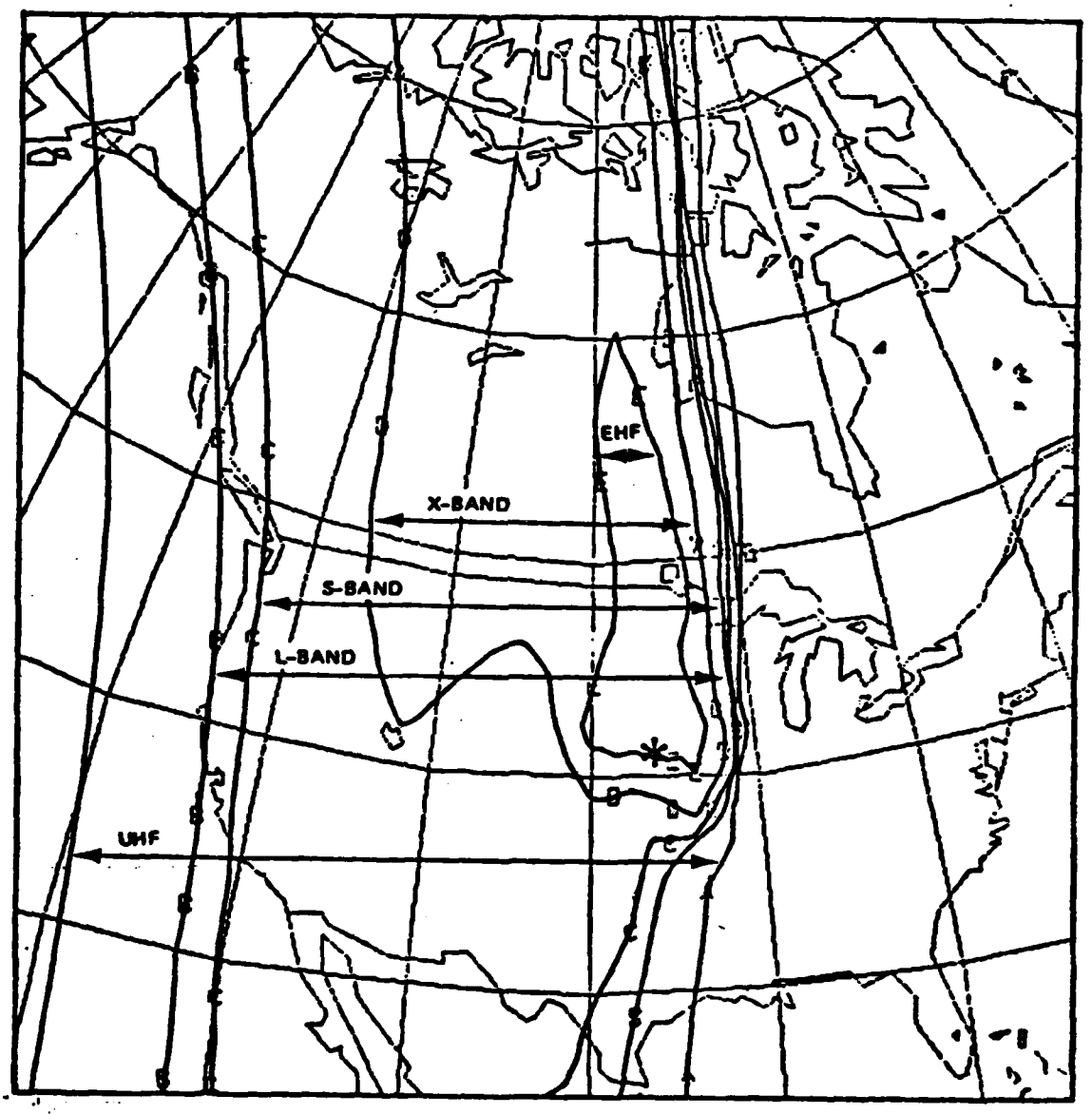


Figure 3. Comparison of the Extent of Intense Scintillation as a Function of Frequency.

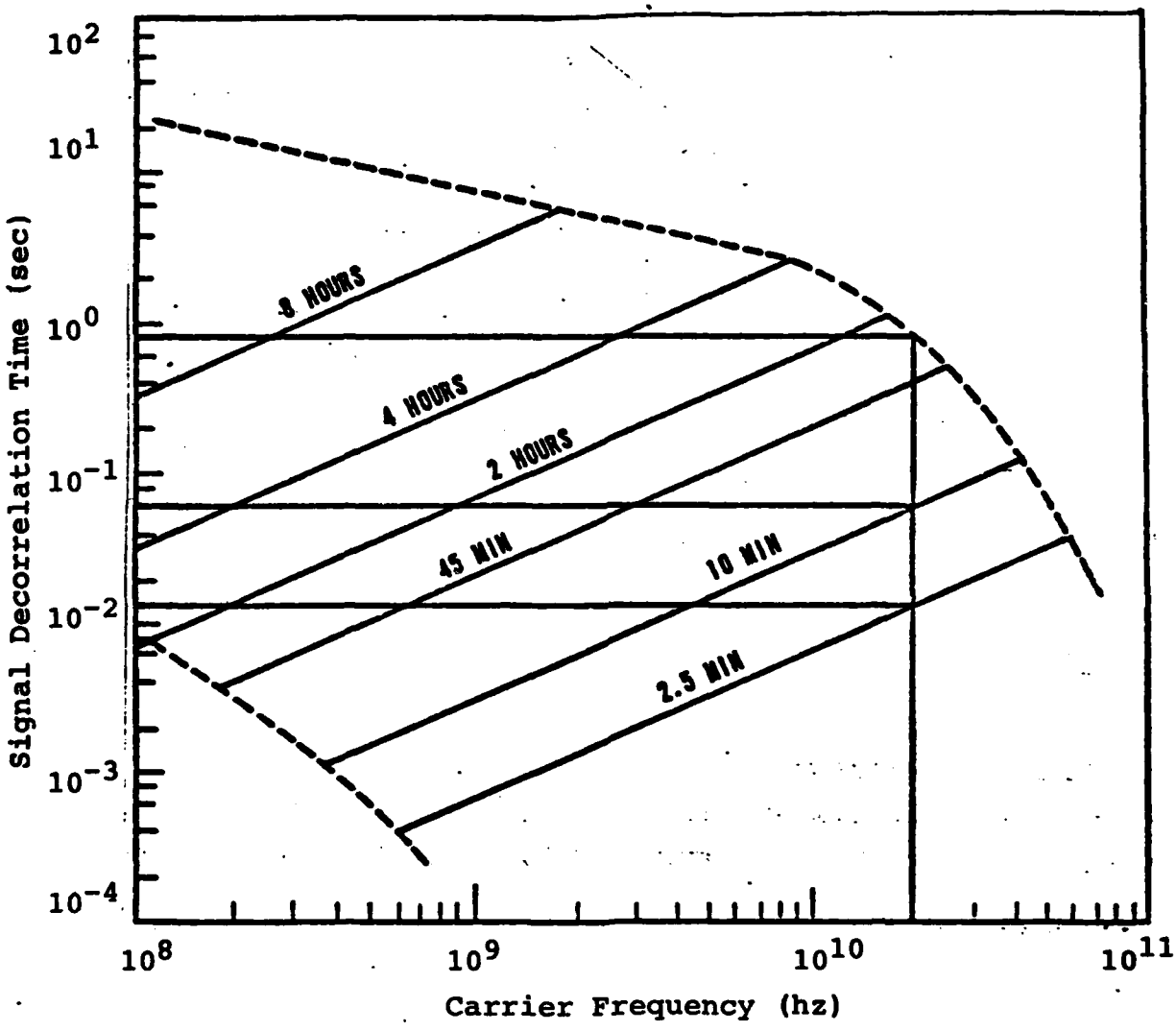


Figure 4. Range of Signal Decorrelation Times for Various Frequencies.

a fundamental rate characteristic of the demodulator. This fundamental rate characteristic is the data rate or code symbol rate at which the RF carrier wave is being modulated. Receiver degradation of this kind is primarily due to the phase scintillation effects rather than the scintillated amplitude fading. Fast fading in essence re-modulates the propagated radio wave and subsequently generates demodulated data bit or code symbol errors in the receiver due to lost of phase coherence and/or the spreading of received signal energy over a wider undetectable bandwidth.

The probability of demodulated bit errors (BER) improves as the received bit energy to noise ratio (E_b/N_o) increases. In extremely rapid fading, τ_o = 0.004 sec , for example, the BER improvement slows down and eventually levels of at some maximum value of E_b/N_o. This leveling off of BER performance vs E_b/N_o is because phase scintillation is not reduced by increasing the signal power. In fact, the initial BER improvement with increasing E_b/N_o is due to the reduction of the contribution of amplitude fading with increased received signal levels or link margin (Ref 7:146). The demodulated errors, which can occur frequently with low E_b/N_o, are in general memoryless or uncorrelated. It is worthy to note, that the memoryless channel characteristics generalized by rapid phase scintillation is one reason forward error correction coding using convolutional codes and Viterbi or sequential decoding is a useful mitigate.

Amplitude fading and noise are the dominant sources of link degradation at slow fade rates (large τ_0) and low E_b/N_o. This performance reduction is directly interpretable in terms of loss in effective link power margin. In the slow fading channel demodulated symbol errors are no longer uncorrelated but occur in burst or clumps. For the EHF link, slow fading channel conditions are encountered more frequently and over longer periods than faster fading channel conditions. To enable forward-error-correction, convolutional encoded systems using Viterbi or sequential decoding must incorporate some kind of interleaving. Interleaving is used to randomize the errors that are generated by the slow fading channel, and thus improves the effectiveness of convolutional codes in providing a powerful random-error correction capability for the link.

Error-correction coding and interleaving can provide significant mitigation of scintillation fading effects where feasible in terms of bandwidth, power and allowable data through put delays. Previous link assessments at other data rates have shown convolutional codes with Viterbi decoding typically provide improved performance in a disturbed channel (Ref 7:152-164) and (Ref 12:3-52).

III. Description of the Simulated EHF, 2400 bps, DBPSK Link

The performance assessment of sequentially decoded long constraint length codes and EHF, 2400 bps, link in both the additive white Gaussian noise (AWGN) and nuclear scintillated signal environment was performed using detailed digital computer generated link simulations. All of the DBPSK link simulations were done using the AFWL FSK-PSK Link Performance Code. The AFWL code produces software realizations of an actual DBPSK modem or receiver that is typically implemented in digital hardware and software.

One of the most important reasons for using detailed digital simulations is because it is not possible to obtain exact analytical expressions for bit error probability for any phase-shift keyed (PSK) link with convolutional coding and sequential or Viterbi decoding, even for an AWGN channel. Furthermore, in the presence of signal amplitude and phase scintillation disturbances, conventional analytical techniques are hopelessly inadequate to evaluate the performance of PSK links (Ref 3:119). Therefore, the approach taken here entails the use of detailed simulation models of convolutional encoders, and sequential and Viterbi decoders in conjunction with the digital simulation model of the DBPSK modem.

One reason for detailed digital link simulations entails the essential requirement for a nuclear induced scintillated signal structure which; otherwise, would have to be produced

by a high-altitude nuclear detonation. In addition, commercially produced sequential and Viterbi decoders were unavailable. Link simulations also provide the capability to easily change receiver design parameters and received signal parameters, as well as, providing easily accessible and derived statistical link performance assessment results.

The AFWL FSK-PSK Code is a program which simulates the operation of several types of receivers used in frequency-shift keyed (FSK) and phase-shift keyed (PSK) satellite communication links in normal and disturbed propagation conditions. Desired modem design parameters and signal descriptions are inputs to the FSK or PSK modem simulation programs. The resulting output includes the received messages and various other performance measurements for the satellite communications link. A description of the PSK and FSK data files is provided in Appendix A.

Representative DBPSK satellite communication modem processing channels implemented in the link simulations for this investigation are illustrated in Figure 5. In addition, Figure 6 shows in block diagram form the three independent processes employed in performing the channel encoded 2400 bps, DBPSK satellite link simulations. As illustrated by Figure 6, the processes include signal propagation modeling, DBPSK modem modeling, and the link performance analysis.

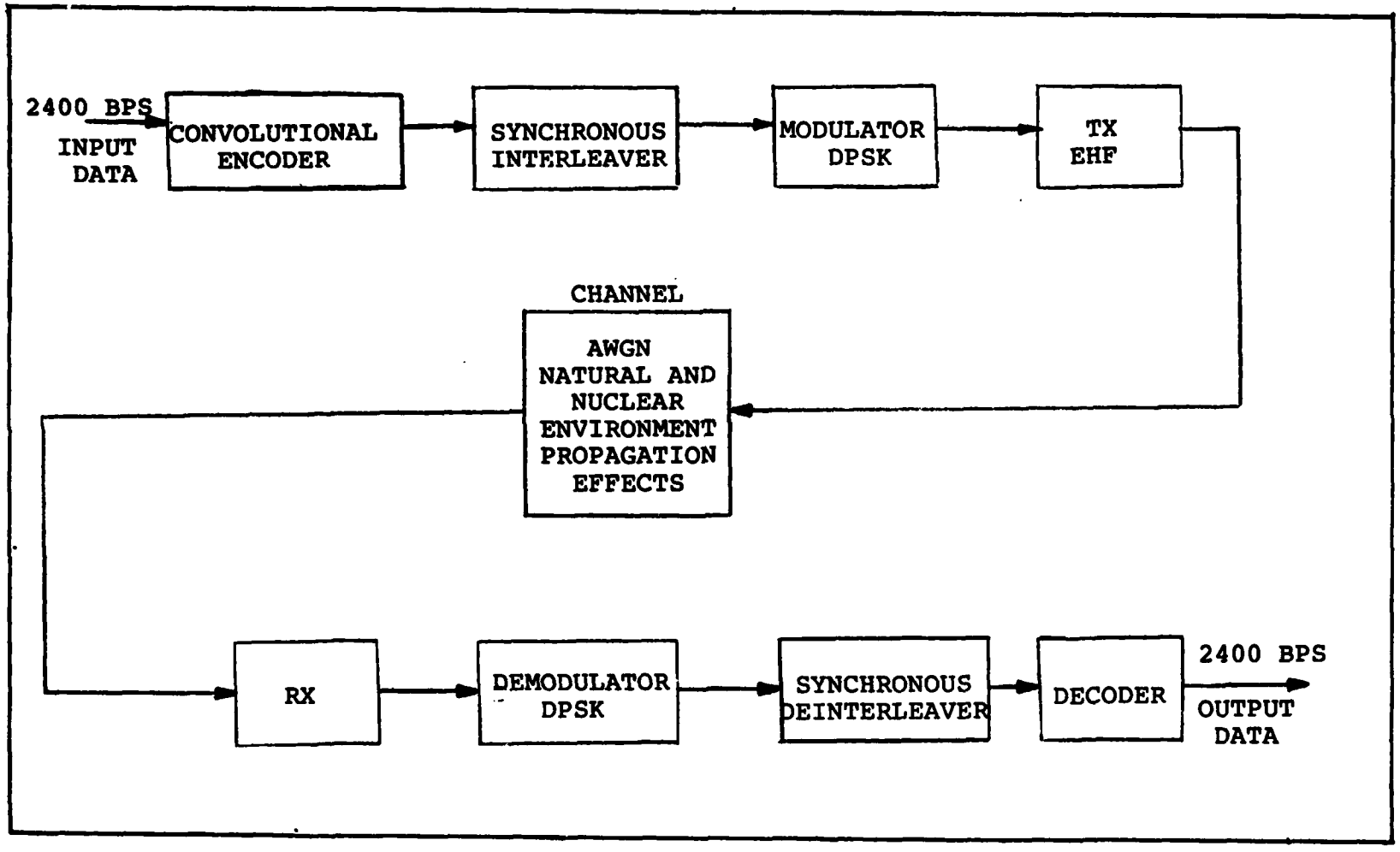


Figure 5. Simulated DBPSK Satellite Communication Modem Processing Channel.

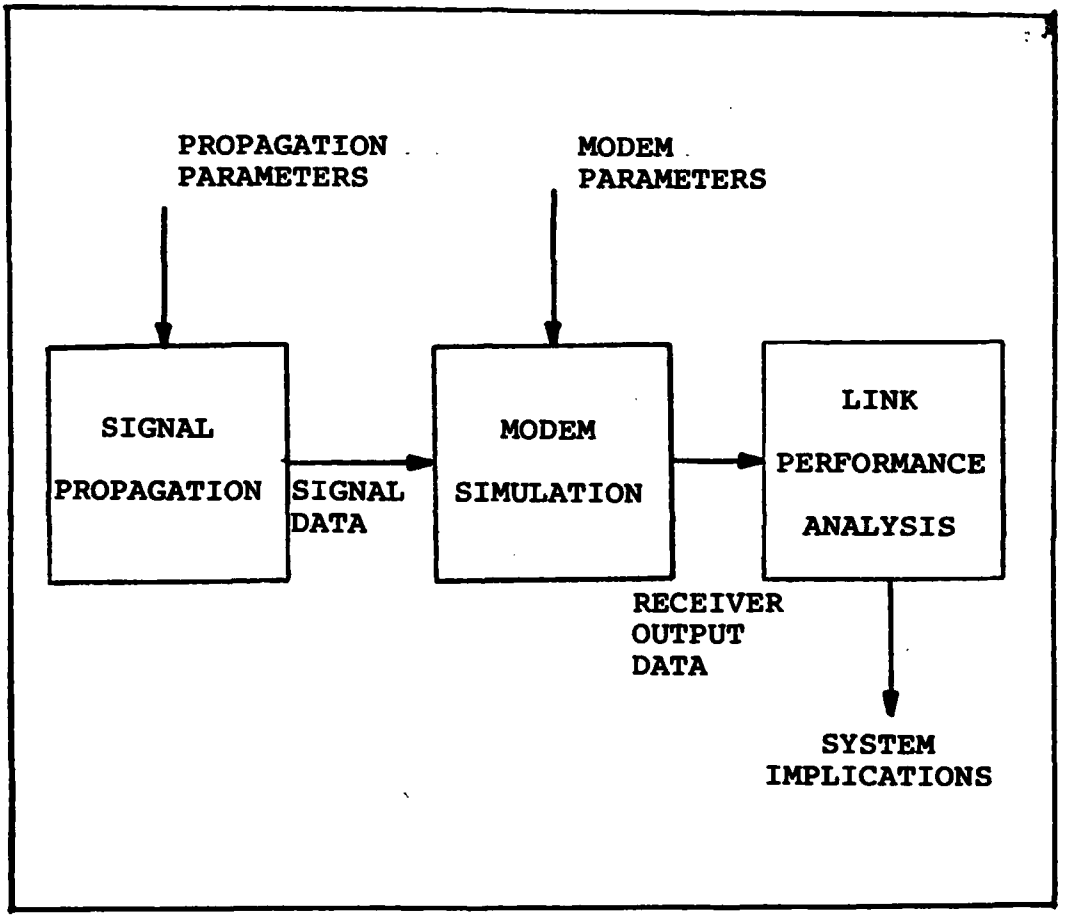


Figure 6. Link Simulation Processes.

DBPSK Modem Simulator

The DBPSK receiver simulator model is a straight forward digital software implementation of differential encoding of the information bit stream, convolutional encoding, interleaving, DBPSK modulation and demodulation, soft-decision quantization of received symbols, deinterleaving and Viterbi or sequential decoding. Within the demodulation section of the simulation, the automatic frequency control (AFC) and automatic gain control (AGC) receiver functions are implemented as well. In addition, the DBPSK simulator interprets, transmits and receives a user generated message, up to 80 characters long, using 6-bit ASCII character text.

Since an actual DBPSK receiver would be implemented primarily in digital hardware or software, the implementation is precise for much of the simulation, with the exception that a general-purpose digital computer, Digital VAX 11/780 used in these link performance analyses, operates with more resolution than is typical of special-purpose hardware (Ref 7:94). Because encoders and decoders are actually implemented digitally, the digital DBPSK modem simulations of these devices are more than a simulation; they are in fact actual devices implemented in a Fortran-coded computer program. The details of the various functions modeled by the DBPSK modem will be explained later in this chapter. The computer simulation models of the convolutional encoders, synchronous interleaver, sequential decoder and Viterbi

decoder that were developed for this investigation or already existed in the AFWL code are described in the following chapters.

Signal Propagation Model

A detailed description of how the scintillated signal structures are generated is beyond the scope of this thesis; however, sufficient documentation exist on the specific characteristics and techniques for modeling a scintillated signal (Ref 28:4). For the purpose of this investigation, it suffices to say time selective (flat fading) nuclear scintillated signal structures are provided to the DBPSK simulator to induce the degraded modem performance discussed in Chapter II.

The signal propagation model consists of a set of data files which represent scintillated signal structure realizations that were calculated using a detailed numerical propagation method called the multiple phase screen (MPS) technique. Outputs of the MPS calculations provide detailed predictions of signal amplitude and phase scintillation caused by the relatively small-scale ionization irregularities in striated regions along the propagation paths. The resulting data file or MPS realization contains the signal amplitude and phase fluctuation samples needed to generate the desired scintillated signal fading period, τ_0.

These scintillated signal amplitude and phase fluctuation samples are inputted directly into the DBPSK modem

simulator by way of a signal sampling routine in the AFWL FSK-PSK Code. These samples define the signal structures present at the input to the receiver. The DBPSK modem simulation model remains completely independent of the technique used to obtain the desired signal amplitude and phase characteristics for a particular link simulation.

White Gaussian noise is added to the signal components that are produced in the demodulator prior to baseband processing. AWGN samples are always injected into the quadrature components of the demodulated signal whether the link simulation is being performed for the normal or disturbed propagation channel. The relationship between the demodulated signal components and the AWGN samples will become more evident in a later section on the DBPSK demodulator. Each AWGN sample consist of a pair of independent Gaussian variates generated by a subroutine in the AFWL Code.

Link Performance Analysis

Results from the detailed digital simulation of the DBPSK modem are input to the link performance analysis section. Performance data is outputted as functions of important signal parameters (τ_0 and E_b/N_0). The link analysis output includes the actual received messages, demodulator and decoder error summary, statistical error rates for uncoded DBPSK, frequency tracking error statistics, measured signal statistics, and for sequential decoder simulations, the number of data bit computations. Refer to examples in Appendix B.

Satellite communications link performance measurement of most interest for the DBPSK modem simulations with Viterbi decoding is the received decoded bit error rate (BER). The BER as well as the number of bit computations are the significant measurement values for the link performance involving sequential decoding.

Differentially Coherent Binary Phase-Shift Keying (DBPSK)

Phase-shift keying (PSK) is a technique for transmitting digital information, '1''s and '0''s, over a radio frequency (RF) communications channel. As shown in Figure 7, the digital sequence is usually represented as a signal consisting of positive and negative voltages. Binary '1''s and '0''s are represented by +1 volts and -1 volts respectively. This baseband signal modulates a RF carrier wave so as to change the phase of the carrier signal upon the transition of a '1' to a '0' or vice versa.

There are two binary PSK modulation schemes commonly used to transmit digital data; coherent binary PSK (BPSK) and differentially coherent binary PSK (DBPSK). Both methods refer to the technique used at the receiver to demodulate a data stream. In both cases, a digital sequence is demodulated by establishing some kind of phase reference with the received modulated signal. BPSK requires the acquisition of a local reference signal in phase coherence with the received carrier; whereas, DBPSK obtains a phase reference by comparing the

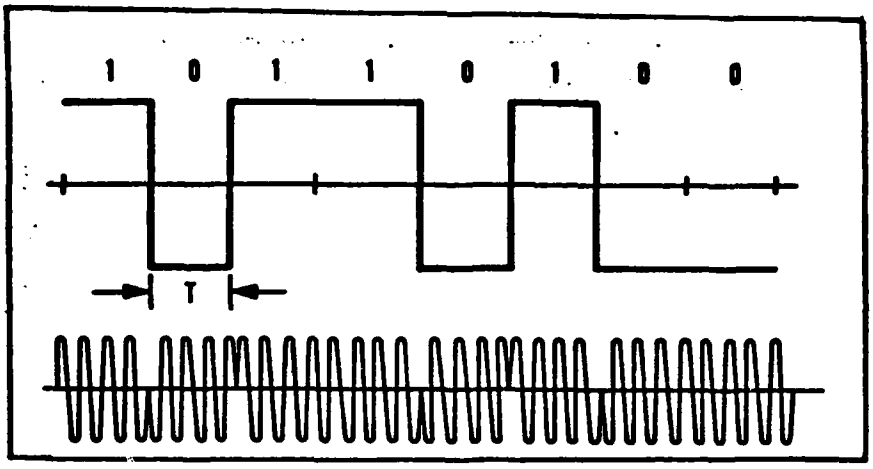


Figure 7. Example of DBPSK Modulated Waveform.

TABLE I

Differential Encoding/Decoding Example

Message sequence:		1	0	0	1	1	1	0	0	0
Encoded sequence: Reference digit	1	1	0	1	1	1	1	0	1	0
Transmitted phase:	0	0	π	0	0	0	0	π	0	π
Transmitted encoded+ bipolar signal:		+	-	+	+	+	+	-	+	-
Received encoded: sequence	1	1	0	1	1	1	1	0	1	0
Received sequence: delay one bit		1	1	0	1	1	1	1	0	1	0
Decoded sequence: (Rx seq.) X (Rx delayed seq.)		1	0	0	1	1	1	0	0	0

carrier phase of the current signaling interval with that of a previous signaling interval. Thus, a DBPSK receiver uses a demodulation technique which detects a relative change in phase from one signaling interval or bit period to the next. A decision of a '1' or '0' is then made based upon the resultant demodulated output signal amplitude or polarity.

The implementation of DBPSK scheme requires: (1) the mechanism causing an unknown phase perturbation on the signal varies so slowly that the phase is essentially constant from one signaling interval to the next. (2) The phase during a given signaling interval bears a known relationship to the phase during the previous signaling interval. Whether the former requirement is satisfied depends, for example; on the stability of transmitter oscillator and the receiver's local oscillator, signal phase changes induced by various disturbances, and so on.

The latter requirement is met by employing differential encoding of the binary message sequence of the transmitter. Table I shows an example of differential encoding and the resultant demodulated sequence. Differential encoding is a match-mismatch encoding process between the message sequence and the encoded sequence. If the new data bit in the message sequence is equal to the last encoded bit, the new encoded bit becomes a '1'. In the case of a mismatch, the encoded bit will be a '0'. Differential encoding of the 2400 bps

data was incorporated in all of the link simulations performed in this investigation.

The block diagram of Figure 8 illustrates the generation of DBPSK. The negation of an EXCLUSIVE-OR logic circuit performs the differential encoding operation. Following, a level shift at the output of the logic circuit so that the encoded message becomes bipolar, the DBPSK signal is produced by multiplication with the carrier. The resultant transmitted DBPSK signal is a double sideband modulated signal.

Signal and Noise Representations for the DBPSK Receiver

This section describes the received signal for the DBPSK demodulator. The AFWL FSK-PSK Code utilizes a digital sampled data receiver structure, illustrated in Figure 9, in which the DBPSK demodulator is implemented as a quadricorrelator. The development below begins with the mathematical representation of incoming signal plus noise and proceeds to the decision variables for the DBPSK demodulator. Analytical expressions for the signal and noise within the demodulator are given in terms of received signal parameters and noise power spectral density. The quadricorrelator signal and noise presentation is the same as developed in an AFWL technical report on signal propagation effects on various satellite systems (Ref 7:47-96).

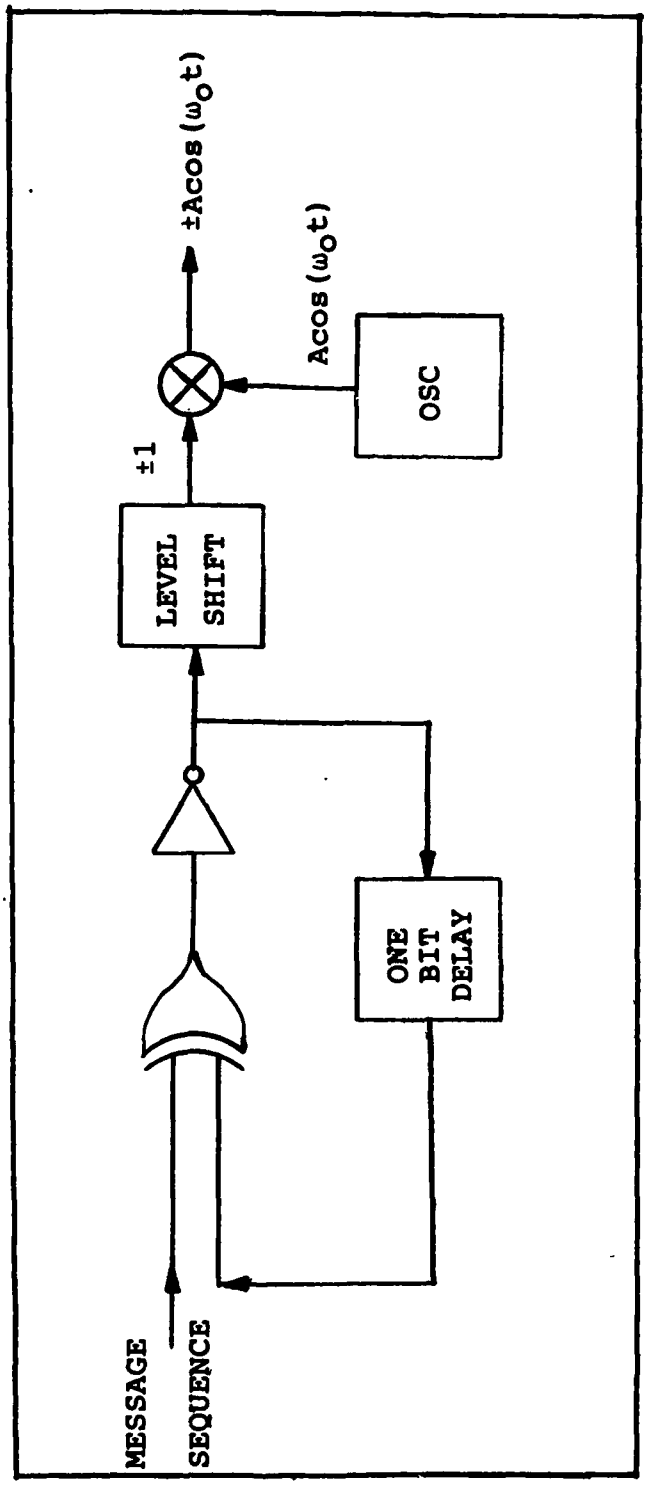


Figure 8. DBPSK Modulator.

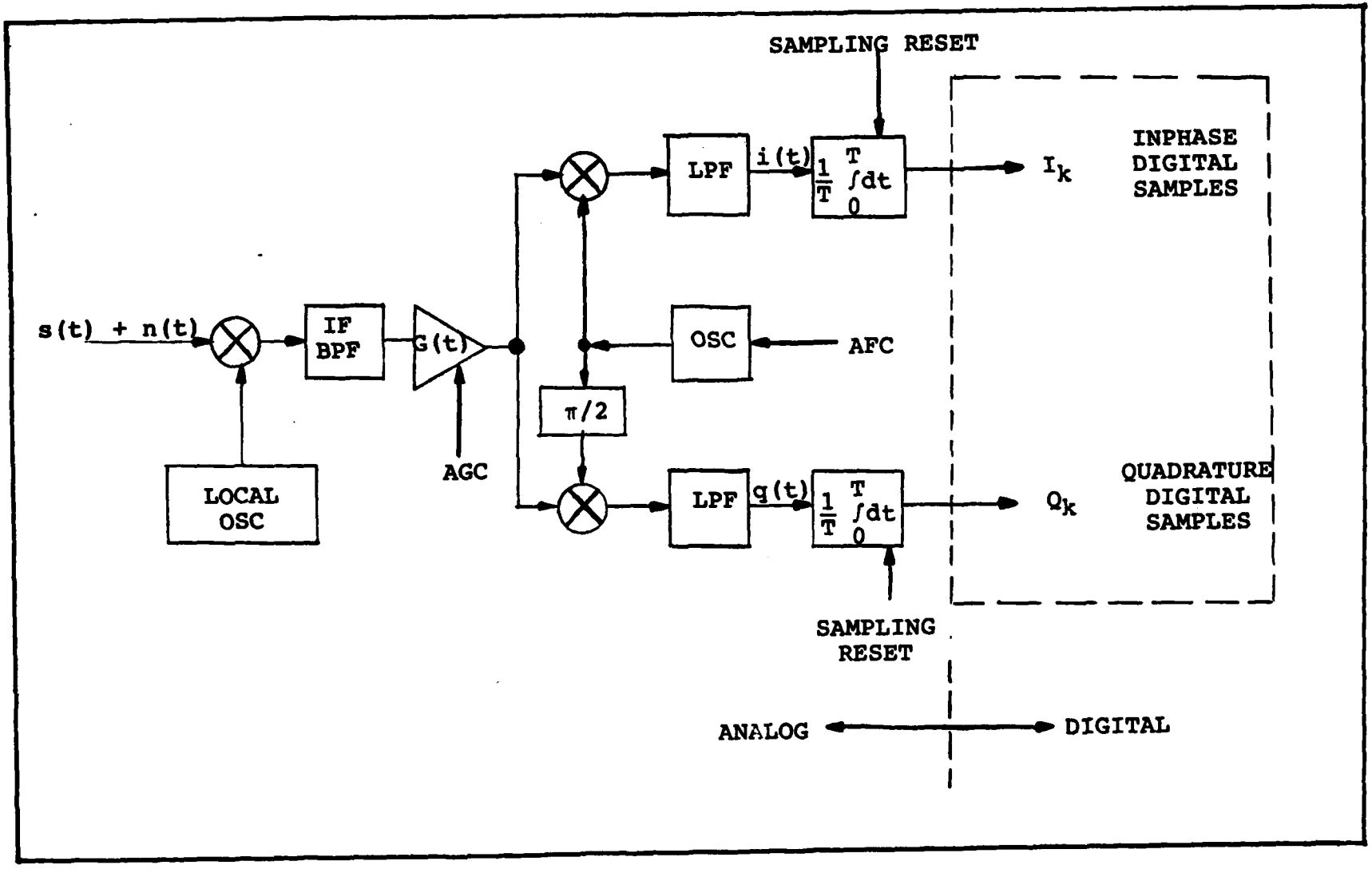


Figure 9. Functional Diagram of Digital DBPSK Demodulator

DBPSK Demodulator. The intermediate frequency (IF) signal at the quadricorrelator in Figure 9 can be written in general form as

$$s(t) = \sqrt{2}\, A(t)\cos[\omega_o t+\theta(t)+m(t)] \tag{2}$$

where

$A(t)$ = signal voltage amplitude

$\theta(t)$ = unmodulated signal phase

$m(t)$ = signal modulation

ω_o = nominal carrier frequency

The factor $\sqrt{2}$ is used for convenience in the following anaylsis.

For DBPSK, the function $m(t)$ represents the binary phase-shift keyed modulation, i.e.,

$$m(t) = 0 \text{ or } \pi \tag{3}$$

corresponding to the two binary data states.

The band-limited (IF filtered) noise may be written as

$$n(t) = \sqrt{2}\, n_1(t)\cos\omega_o t - \sqrt{2}\, n_2(t)\sin\omega_o t \tag{4}$$

$$= \sqrt{2}\, n_m(t)\cos[\omega_o t+\alpha(t)] \tag{5}$$

where $n_1(t)$ and $n_2(t)$ are independent zero-mean Gaussian random processes, and

$$\sigma_{n_1}^2 = \sigma_{n_2}^2 = \frac{N_o B}{2} \tag{6}$$

$$n(t) = \sqrt{n_1^2(t) + n_2^2(t)} \tag{7}$$

$$\alpha(t) = \tan^{-1}\left[\frac{n_2(t)}{n_1(t)}\right] \tag{8}$$

N_o = one-sided noise power spectral density

B = IF filter noise bandwidth

The input signal plus noise is passed through an AGC with gain G(t) and is then translated down in frequency to form two quadrature baseband channels, as shown in Figure 9.

Let the local oscillator output be represented by

$$v(t) = \sqrt{2}\cos[\omega t+\hat{\theta}(t)] \tag{9}$$

where $\hat{\theta}(t)$ is the local oscillator phase relative to the nominal carrier phase $\omega_o t$.

The two quadrature baseband voltages are the lowpass-filtered outputs obtained by mixing the IF signal with the local oscillator:

$$i(t) = G(t)[s(t)+n(t)]\ \sqrt{2}\cos[\omega_o t+\hat{\theta}(t)] \tag{10}$$

$$q(t) = G(t)[s(t)+n(t)]\ \sqrt{2}\cos[\omega_o t+\hat{\theta}(t)+\tfrac{\pi}{2}] \tag{11}$$

Retaining only difference frequency components, the two baseband voltages are

$$i(t) = G(t)\{A(t)\cos[\phi(t)+m(t)] + n_i(t)\} \tag{12}$$

$$q(t) = G(t)\{A(t)\sin[\phi(t)+m(t)] + n_q(t)\} \tag{13}$$

where

$$\phi(t) = \theta(t) - \hat{\theta}(t) \tag{14}$$

$$n_i(t) = n_1(t)\cos\hat{\theta}(t) + n_2(t)\sin\hat{\theta}(t) \tag{15}$$

$$n_q(t) = n_2(t)\cos\hat{\theta}(t) - n_1(t)\sin\hat{\theta}(t) \tag{16}$$

The quadrature noise voltages $n_i(t)$ and $n_q(t)$ are independent zero-mean Gaussian random processes and have the same variance as $n_1(t)$ and $n_2(t)$:

$$\sigma_{n_i}^2 = \sigma_{n_q}^2 = \frac{N_o B}{2} \tag{17}$$

In each channel the signal is sampled by an integrate-and-dump sampler at intervals of T seconds:

$$I_k = \frac{1}{T} \int_{(k-1)T}^{kT} i(t)dt \tag{18}$$

$$Q_k = \frac{1}{T} \int_{(k-1)T}^{kT} q(t)dt \tag{19}$$

where k is a discrete time index. This results in the following digital samples

$$I_k = G_k[A_k\cos(\phi_k+m_k) + x_k] \tag{20}$$

$$Q_k = G_k[A_k\sin(\phi_k+m_k) + y_k] \tag{21}$$

where for simplicity in these expressions the AGC gain, signal phase, and signal level have been approximated by constants over the sample interval. The noise samples x_k and y_k are uncorrelated zero-mean Gaussian variables with variance

$$\sigma_x^2 = \sigma_y^2 = \frac{N_o}{2T} \tag{22}$$

The sampled waveform of Eqs (20) and (21) has the modulation component $m_k = 0$ or π, which, as previously mentioned, can be viewed as bipolar amplitude modulation.

With differentially coherent demodulation, the I_k and Q_k samples are first accumulated over a data bit period if the receiver sampling rate exceeds the bit rate:

$$I_K = \frac{1}{K} \sum_{k=1}^{K} I_k \tag{23}$$

$$Q_K = \frac{1}{K} \sum_{k=1}^{K} Q_K \tag{24}$$

where K is the number of samples per bit. A similar accumulation of samples over a preceding bit provides the reference for DBPSK demodulation. The binary decision variable is obtained by a "dot product" of the current sample pair and the reference samples:

$$D = I_K I_R + Q_K Q_R \tag{25}$$

where I_R and Q_R are the reference samples. The binary decision is given by the polarity of D .

In the 2400 bps, rate 1/2 encoded, DBPSK link simulations, the I_K and Q_K samples were actually accumulated over a code symbol period corresponding to:

$$T_C = \frac{T_D}{R} \tag{26}$$

where

T_D = data bit period = 1/2400 sec

= 1/2400 sec

R = convolutional encoding rate = 1/2

= 1/2

i.e.,

T_C = 1/4800 sec

= .2083 ms

The receiver A/D sampling rate used in the simulations was equal to the Nyquist sampling rate or twice the modulational symbol rate given by:

$$F_S = \frac{2}{T_C} \tag{27}$$

= 9600 Hz

thus, the number of samples per received code symbol was, $K = 2$.

In addition, the demodulated code symbols in the link simulations were quantized upon the polarity and amplitude value of D in Eq (25).

Finally, the automatic frequency control (AFC) and automatic gain control (AGC) will not be presented in this document but are described in detail in other references (Ref 7:69-72, 179-188) and (Ref 5;119-129, 165-171). AGC and AFC types and corresponding parameters used in the DBPSK link simulations are shown in the sample outputs of Appendix A.

IV. Convolutional Codes

Convolutional Encoding

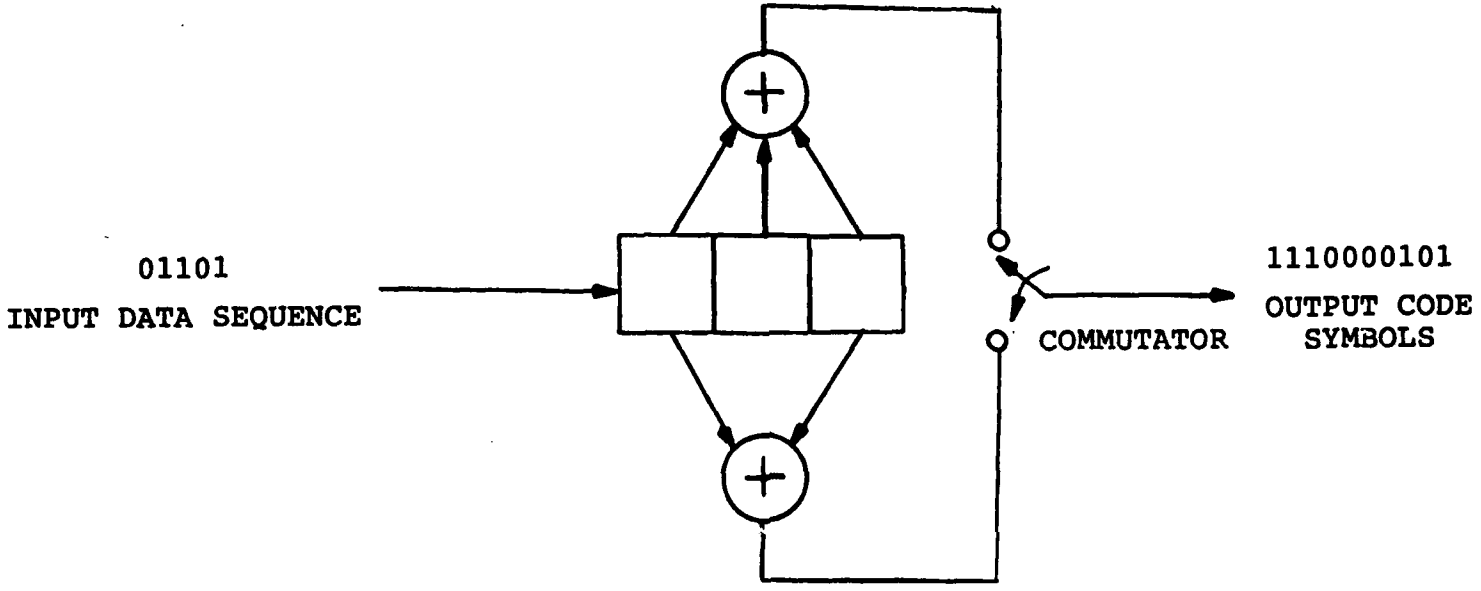
There are two requirements for the use of convolutional encoding in the digital satellite link simulations. The first requirement simply entails the necessary channel encoding data. An input data stream is shifted into the convolutional encoder and the output from this encoder is used to modulate an RF carrier for subsequent simulation of the propagation, demodulation and decoding processes. Second, the Viterbi and sequential decoders at the receiver utilize an encoder of identical configuration to that used in the encoding process.

The convolutional encoder is commonly described as a linear finite state machine consisting of a multi-stage (bK) binary shift register and a specified number (v) of modulo-two adders. Each adder is connected to certain stages of the binary shift register. Input data bits are shifted into and through the shift register b bits at a time. After each b-shift, the modulo-two adders are sequentially sampled yielding v code symbols. At a particular sample time, each code symbol which is generated depends on the current contents of the encoder shift register and the pattern of connections between the shift register stages and the modulo-two adder generating that code symbol. As v code symbols are generated, they are sequentially used to modulate a RF

carrier. The code rate R, is the ratio of input information bits to output symbol bits. Thus $R = b/v$ where $b < v$.

The code constraint length, K, is the number of b bit shifts over which a single information bit can influence the encoder output. The state of the encoder can be specified, at any time, by the contents of the first, $b(K-1)$ shift register stages, and the number of possible states is $2^{b(K-1)}$. The number of possible transitions from any one state to other states depends on the number of input data bits (b) shifted into and through the shift register at a time, specifically 2^b transitions.

Figure 10 shows a binary-input, binary-output convolutional encoder consisting of a three stage binary shift register with two modulo-two adders where $b = 1$, $K = 3$ and $v = 2$. It is assumed that the encoder is initially in the zero state, i.e., the shift register is initially cleared. As the first information bit is inserted in the left-most shift register stage, the shift register contents are shifted to the right one stage. As a result of the first input bit in the data sequence (a '1' at $t = 1$), the encoder state as specified in the first two stages reflects a transition from state 00 to state 10. The output code symbol set generated by the modulo-two adders are two '1' bits, which are used sequentially to modulate the carrier. As indicated by Figure 10, subsequent input data bits are



DATA BIT NUMBER	INPUT DATA SEQUENCE	ENCODER STATE	SHIFT REGISTER CONTENTS	OUTPUT CODE SYMBOL SET
0	-	00	000	--
1	1	10	100	11
2	0	01	010	10
3	1	10	101	00
4	1	11	110	01
5	0	01	011	01

Figure 10. Convolutional Encoder for a Rate 1/2, Constraint Length 3 Code.

shifted into and through the encoder shift register which in turn generate additional code symbol bits.

Figure 11 is the tree diagram representation of the convolutional encoder shown in Figure 10. The vertical members in this tree are referred to as nodes and correspond to the time intervals an information bit is inputted into the encoder. The horizontal members are branches and denote the output code symbol bits for that position on the tree. By convention, at each node location in the tree, one moves down if the input bit is a one, and up if it is a zero. Therefore, if the first input bit is a zero, the resulting output code symbols are those shown on the first upper branch, and the encoder has made a transition to state 00, designated 'a' in Figure 11. On the otherhand, if the first input bit is a one, the encoder makes a transition to state 10, designated 'b' and these output code symbol bits are shown on the first lower branch. Similarly, if the second input bit is a zero, the tree diagram is traced to the next upper branch, while if it is a one, the diagram is traced downward. Thus, all 32 possible paths representing the state transitions and output symbol bits associated with each path can be traced for each of the 32 permutations of the first five input data bits. The dashed line in Figure 11 indicates the path for the particular input sequence shown in Figure 10.

Note in Figure 11 after the first three branches, the structure of the tree becomes repetitive. Specifically,

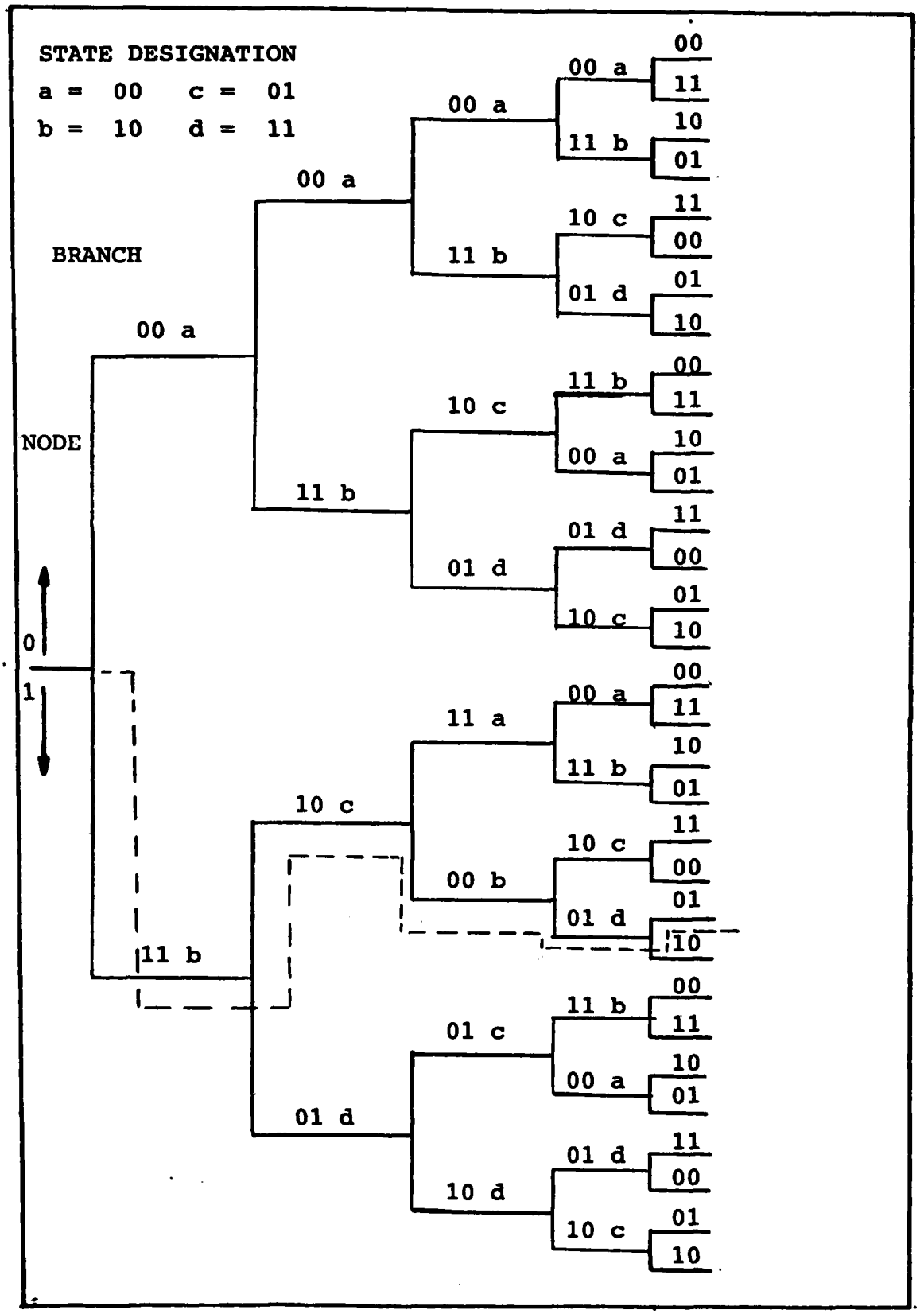


Figure 11. Code Tree for the Convolutional Encoder of Figure 10.

beyond the third branch, the code symbols on branches coming from any given state (a, b, c, d) are identical, and the state transition sequence is the same. This occurs because as the fourth input data bit is shifted into the first (left-most) stage of the encoder, the first input data bit no longer has any effect on the encoder output symbols. Therefore, any input data sequence will create identical code symbol sequences after the third branch.

The trellis diagram, Figure 12, is frequently used to show the encoder output given an input sequence. The name is appropriate because of its tree-like structure with emerging branches. By adopted convention, the code branches produced by a zero input information bit are shown as solid lines and code branches produced by a one input bit are shown dashed. The following example demonstrates how the trellis diagram can be interpreted. If the register is in state 00 and a one is input, the next state is 10. From 10 a zero input would produce state 01, etc. The path through the trellis diagram for the data sequence of Figure 10 is shown in Figure 13.

Another important diagram that is used to demonstrate the properties of any particular encoder is the state transition diagram. This type of diagram is developed by using the properties of the encoder in terms of the current state (first b(K-1) stages), the immediately preceding state (last b(K-1) stages), the input data bit(s) causing the state transition bit(s) located in the first b stage(s)

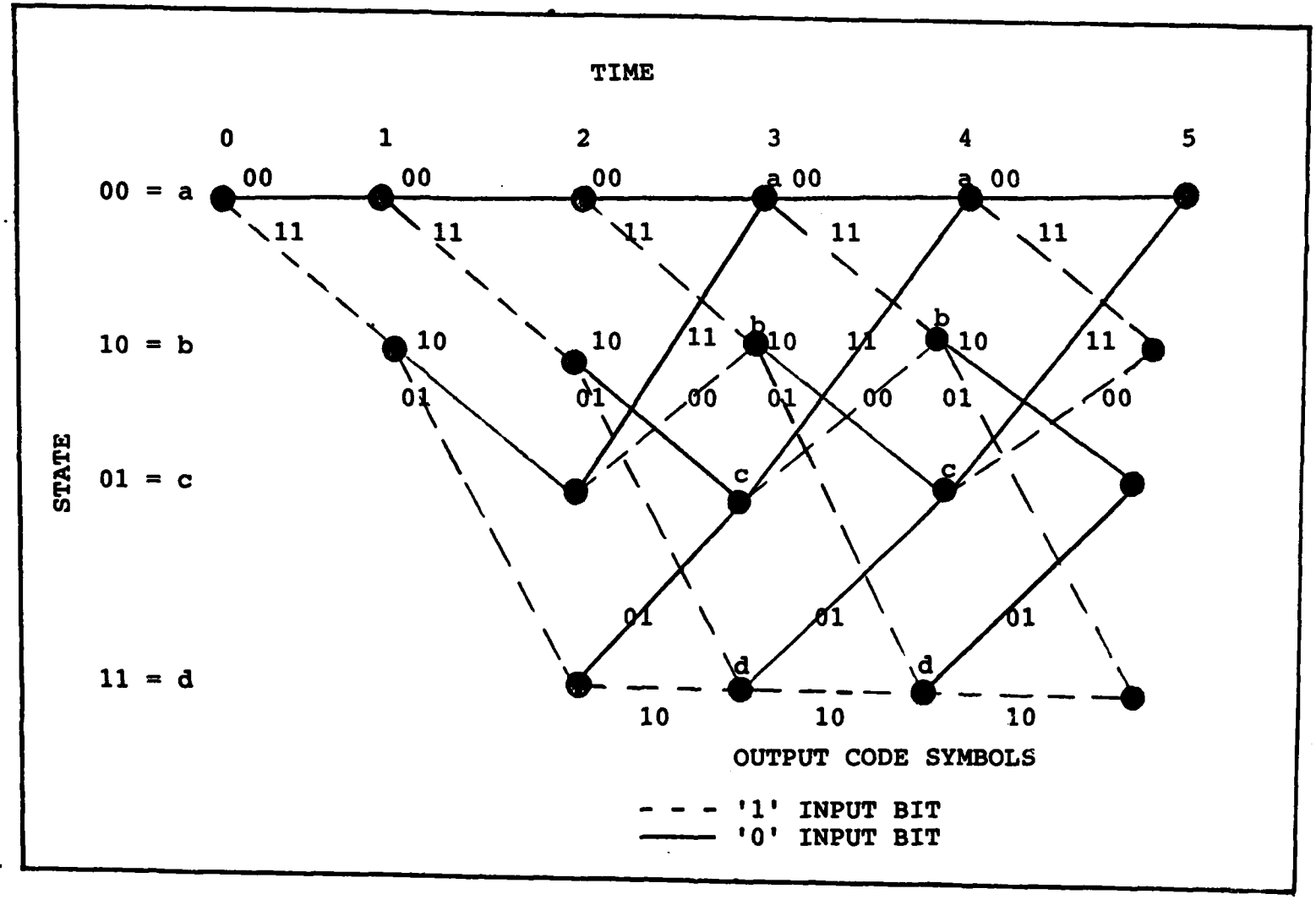


Figure 12. Trellis Code Representation for Encoder of Figure 10.

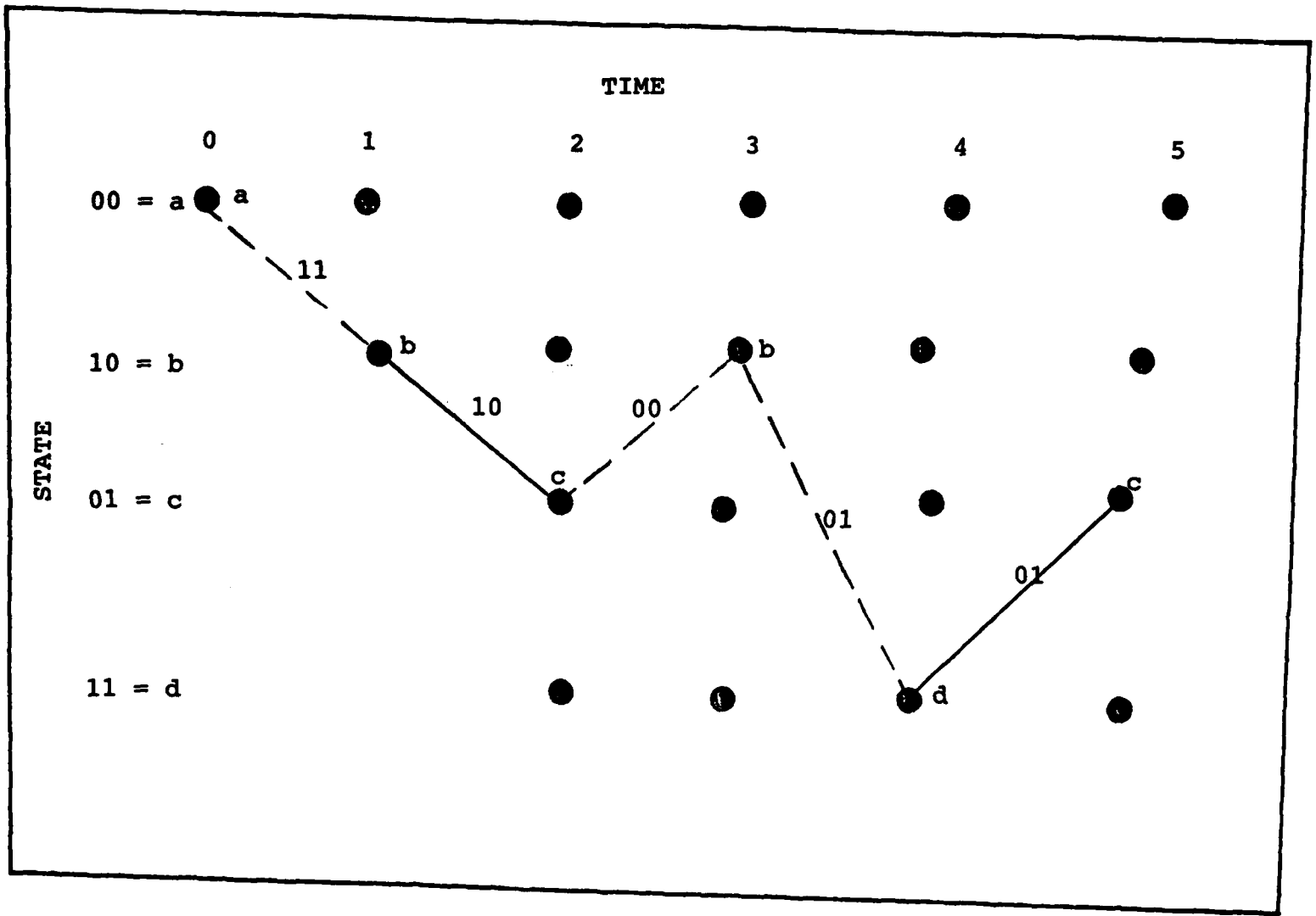


Figure 13. Trellis Path for the Data Sequence of Figure 10.

and the resulting code symbol bits generated by each modulo-two adder operating on the appropriate binary shift register contents. This can best be seen through an example. Referring to Figure 14, if the contents of the shift register are 100 as shown at t = 1 , this indicates a transition to state 10 from the previous state 00 which was caused by an input data bit '1' and produced the two output symbol bits, 11. As a result, Figure 14 shows the state transition diagram segment connecting nodes a and b . In a similar manner, if the shift register contents are 010, as indicated in Figure 10 at t = 2 , this reflects a transition to state 01 from state 10, caused by an input data bit '0' and producing code symbol bits of 10. This can be seen in Figure 14 as the branch connecting nodes b and c .

Characteristics of Convolutional Codes

A convolutional code is often characterized by a measure of code "goodness" known as its minimum free distance. The minimum free distance, d_F , is defined as the minimum number of encoded symbols in which any two arbitrarily long encoded sequences differ (Ref 23:205). The larger the minimum free distance the more powerful the error correction capability. Convolutional codes used in the Viterbi decoding and sequential decoding simulation have large free distances.

Convolutional codes are also characterized as systematic or nonsystematic. This characteristic is determined by the

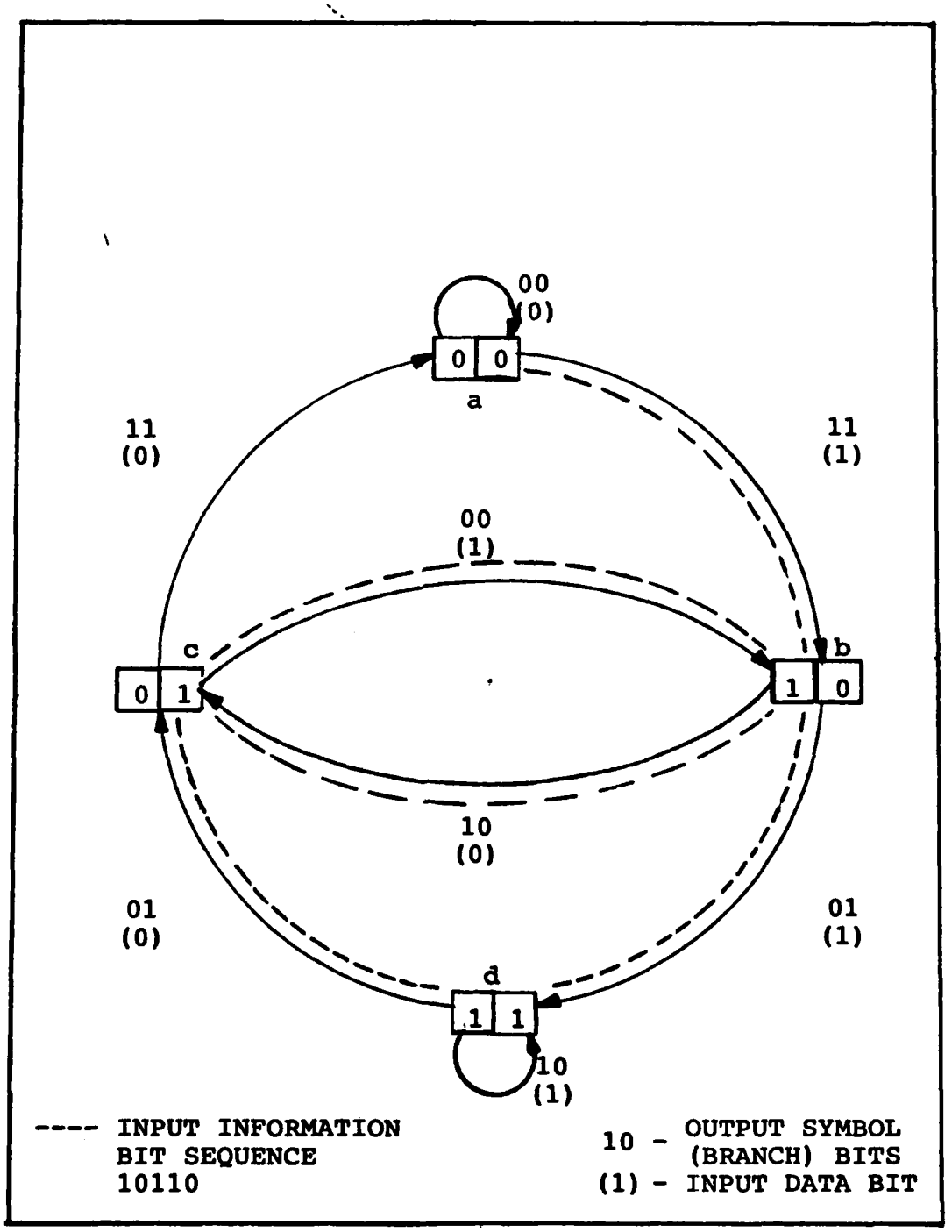


Figure 14. State Transition Diagram for the Convolutional Encoder of Figure 10.

pattern of connections between the modulo-two adders and the stages of the encoder binary shift register. A systematic code produces an output sequence from one modulo-two adder which is a replica of the input data stream. The principle advantage of systematic codes is that the information sequence is easily recoverable from the encoded sequence. Nonsystematic codes, in general, have an inherent superiority in undetected decoded bit error probability over systematic codes when used with sequential and maximum likelihood decoders. Costello (Ref 8:806-813) has shown that for a given constraint length, K , more free distance can be obtained with nonsystematic codes than systematic codes. In addition, the computational behavior of nonsystematic codes with sequential decoding is identical to that of the best systematic codes.

The final code characteristic to be discussed is that of a catastrophic code. A catastrophic code will allow a finite number of channel symbol errors, outputted from the demodulator, to cause an infinite number of data bit errors to be decoded (Ref 22:28-31). To generate a desirable noncatastrophic code at least one of the encoder's modulo-two adders should have an uneven number of connections.

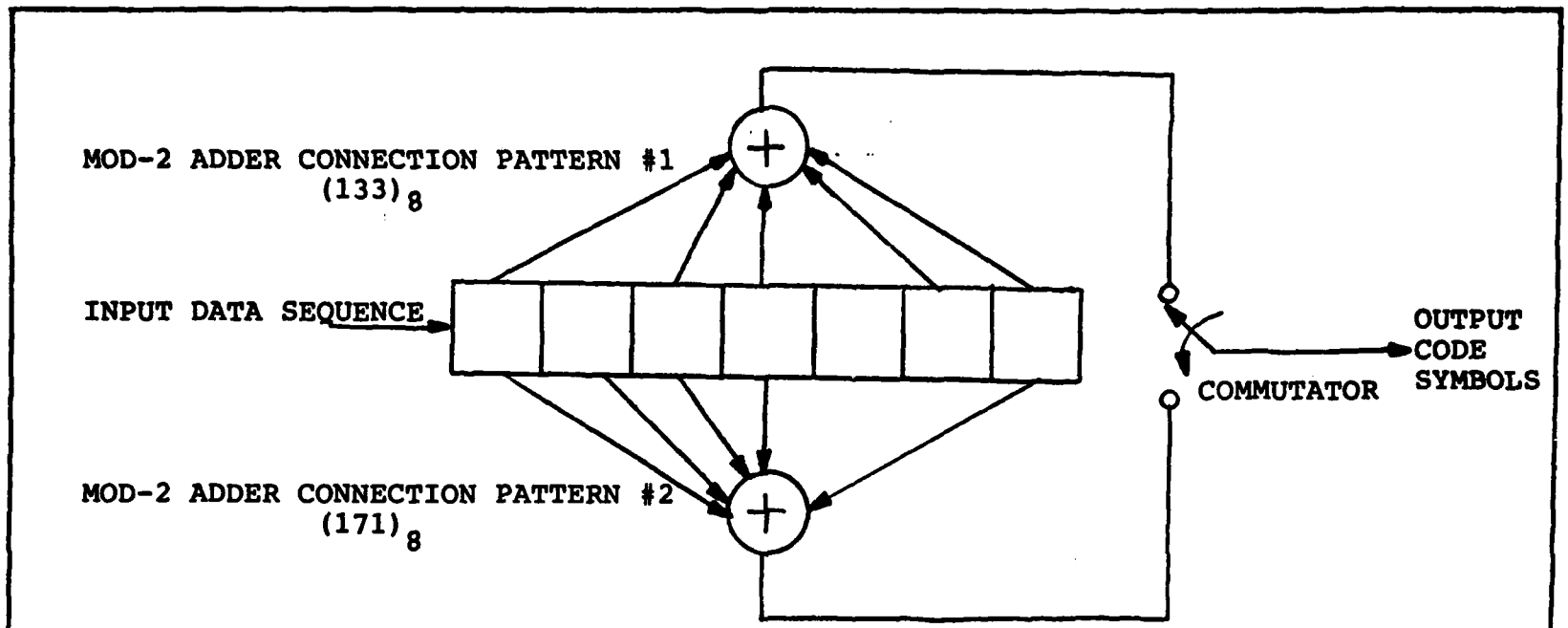
The convolutional codes used in the satellite simulations incorporating Viterbi decoding and sequential decoding are nonsystematic and noncatastrophic.

Convolutional Code Used for the Viterbi Decoding Simulations

As specified by the AFWL thesis sponsor a rate 1/2, constraint length 7 convolutional code was used for all the simulations entailing Viterbi decoding. This is a popular code used in several satellite communication systems. A rate 1/2, constraint length 7 code is one in which each input data bit causes two binary code symbols to be transmitted and affects seven consecutive pairs of encoder output symbols. The encoder configuration for this code is given in octal notation as $(133, 171)_8$. In addition, the code has a minimum free distance of 10, which is the maximum value that can be obtained for a rate 1/2, constraint length 7 convolutional code (Ref 21:371-372).

Octal notation of the encoder configuration is used to describe the connection patterns of the 2 modulo-two adders with the seven binary shift registers shown in Figure 15. The notation represents a binary pattern for each modulo-two adder in which a 1 denotes a connection and a 0 denotes no connection to the corresponding stage of the shift register.

An illustration of the manner in which the code is generated by the encoder is shown in Figure 15. The figure shows the output code symbols that result for the given data bit sequence. In this example, the encoder has an initial state of zero. This example is similar to the one used in Bogusch's DSCS II report (Ref 3:3-121).



DATA BIT NUMBER	INPUT DATA BIT	SHIFT REGISTER CONTENTS	OUTPUT CODE SYMBOL BITS
0	-	0000000	--
1	1	1000000	11
2	0	0100000	01
3	1	1010000	00
4	1	1101000	01
5	0	0110100	10
6	1	1011010	01
7	0	0101101	01
8	0	0010110	01

Figure 15. Convolutional Encoder for Rate 1/2, Constraint Length 7 Code.

Convolutional Codes Used for the Sequential Decoding Simulations

Two convolutional codes were used for the sequential decoding simulations performed for the AWGN and nuclear scintillated channels. The first code rate 1/2, constraint length 30 and was produced by an encoder configuration given in octal notation as $(5335336767, 7335336767)_8$. The second code was also rate 1/2 but with a constraint length of 25. It's encoder configuration is given by $(126726757, 166726757)_8$. Encoder one is illustrated in Figure 16 and encoder two is illustrated in Figure 17.

Both of the above codes are based upon a nonsystematic, noncatastrophic, rate 1/2, constraint length 48 convolutional code developed by Massey and Costello (Ref 8:806-813). This code has the following encoder configuration, $(5335336767373553, 7335336767373553)_8$. The Costello code has a large free distance which makes it well suited to sequential decoding.

Massey and Costello demonstrated their encoder configuration could be used to develop other nonsystematic codes with constraint lengths, $K \leq 48$. Codes, like the two implemented in this investigation, were produced by dropping and/or changing the octal terms on the right-hand side of the Costello code until the desired code constraint length was obtained. For example, the $K = 30$ code was produced by dropping 6 octal terms from the right-hand side of the original code.

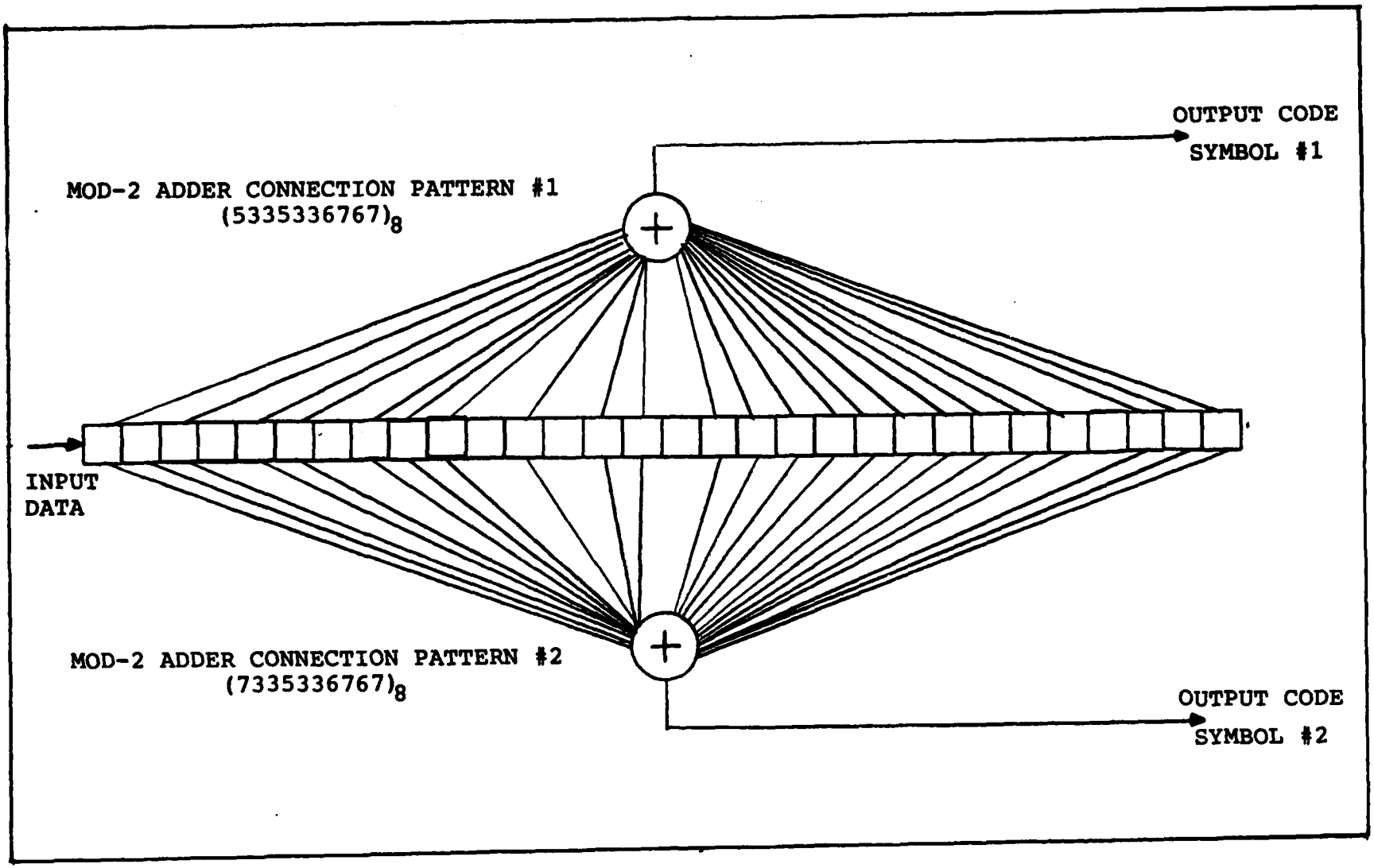


Figure 16. Convolutional Encoder for Rate 1/2, Constraint Length 30 Code.

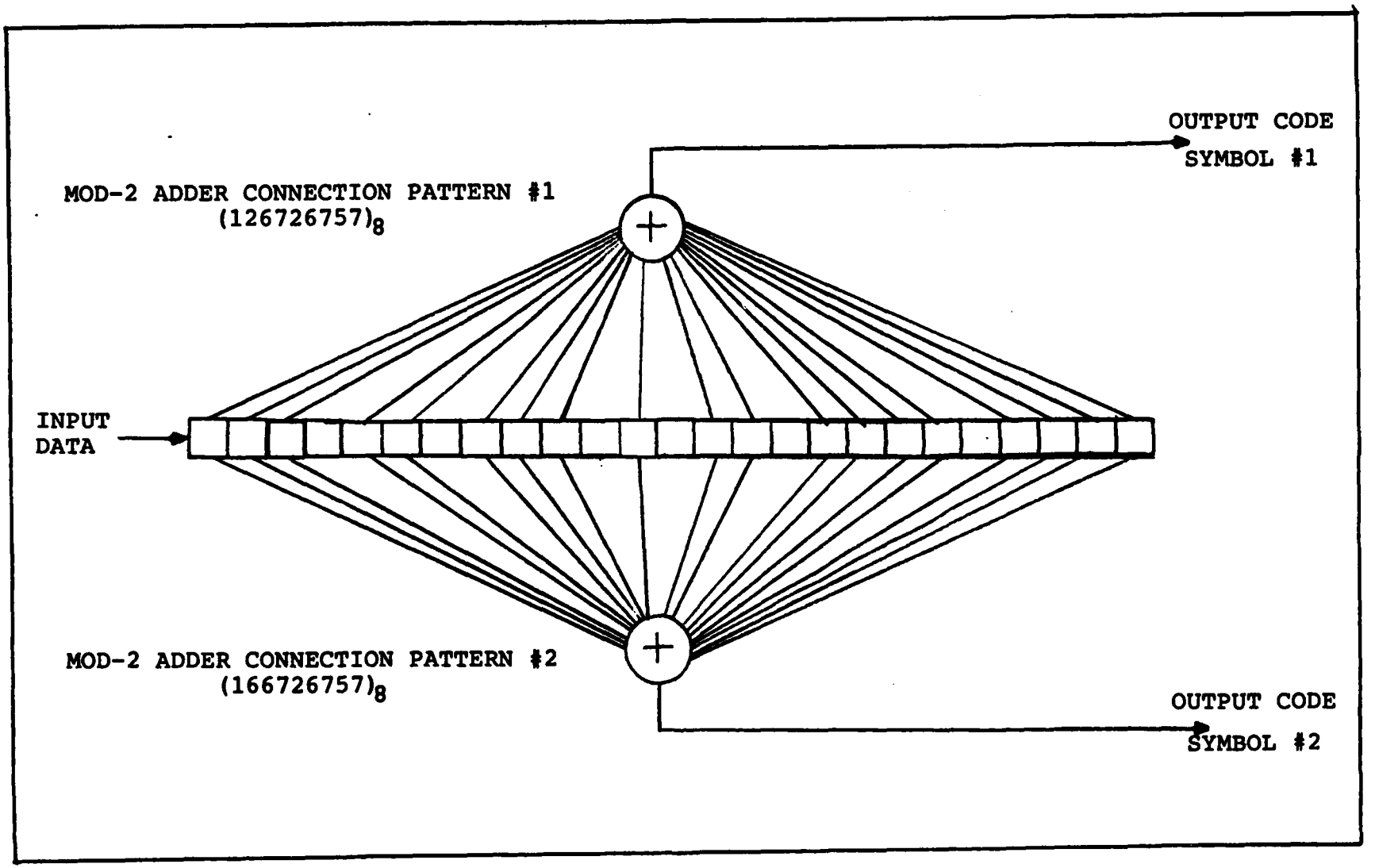


Figure 17. Convolutional Encoder for Rate 1/2, Constraint Length 25 Code.

V. Sequential Decoder Design and Implementation

Sequential Decoding and the Fano Algorithm

Sequential decoding is a procedure for systematically searching through a code tree, using received information as a guide, with the objective of eventually tracing out the path representing the actually transmitted information sequence. Because a sequential decoder is an algorithm for decoding tree codes, convolutional codes which can be represented in the form of a tree are ideally suited for this technique.

The basic task of sequential decoding, to search the code tree and determine a trial information sequence whose encoded version most closely matches in some sense the corrupted received sequence, distinguishes it from other decoding schemes developed for block codes and from other algorithms developed for convolutional codes. For a noisy channel, the trial information sequence or message tree path may not be completely correct over its entire length. However, the sequence has a high probability of being correct at the end corresponding to the information digits about to be decoded. The amount of decoding computation is governed by the level of channel noise.

A measure of the closeness between the trial information sequence and the corrupted received sequence entails the concept of the decoding metric. The simplest decoding metric used for sequential decoding is the Hamming distance. Other

decoding metrics include the mutual information of a particular trial sequence and the received sequence, and sums of quantized integer symbol metrics generated from the soft-quantized output of the demodulator's matched filters. The latter method is used in the developed simulation and will subsequently be described in detail.

Limitation of Sequential Decoding. A fundamental limitation of sequential decoding techniques is the decoder speed required (particularly in real time) to handle extremely noisy received sequences. Such a received sequence typically results in numerous trial searches of the message tree while the sequence is being processed by the decoder. This limitation can be characterized quantitatively in terms of a parameter R_{comp}, the computational cutoff rate. R_{comp} is the maximum rate for which the average amount of computation required to decode a single bit, using sequential decoding, is finite. In addition, R_{comp} is a limit on the computational rate of practical coding systems and has a value less than the channel capacity.

As the channel rate R approaches R_{comp} from below, the computational load increases tremendously. In theory, the average decoding computational effort becomes infinite when $R = R_{comp}$. This, of course, doesn't happen in practice, but a dramatic increase in computation as R exceeds about $0.9\ R_{comp}$ has been observed (Ref 27:153). Therefore, sequentially decoded convolutional codes are only useful at rates less than $0.9\ R_{comp}$.

The parameter R_{comp} is determined in part by the channel noise. For the discrete memoryless channel, DMC, with I input symbols, J output symbols and symbol input probabilities p_i, R_{comp} is given by (Ref 27:153):

$$R_{comp} = \max_{\{p_i\}}\left[-\log \sum_{j=1}^{J} \left[\sum_{i=1}^{I} p_i\sqrt{p(j|i)} \right]^2 \right] \qquad (28)$$

where the $p(j|i)$ are channel transition probabilities and the maximization takes place over all possible sets of input probabilities. For the binary symmetric channel, BSC, where $p_i = 1/2$, $p(j|i) = p$ for $j = 1$, $p(j|i) = (1-p)$ for $j = i$, $J = 2$ and $I = 2$, Eq (28) reduces to

$$R_{comp} = 1 - \log[1+2\sqrt{p(1-p)}] \qquad (29)$$

For the binary input, AWGN channel, we have (Ref 22:38):

$$R_{comp} = \log_2[2/(1+\exp(-E_s/N_o))] \qquad (30)$$

or

$$R_{comp} = 1 - \log_2(1+\exp(-E_s/N_o)) \text{ bits/symbol} \qquad (31)$$

where E_s is the energy per channel symbol, given by $E_s = E_bR$ for the energy per data bit E_b and Rate R in bits per symbol.

Thus, for the AWGN channel, R_{comp} depends on the average properties of the channel (through N_o) and on the transmitted energy. It is important to note that the received

signal energy E_s actually depends, in addition, on properties of the transmission medium. Some of these include: $1/r^2$ attenuation, absorption, dispertion, refraction, multipath and natural/nuclear scintillations.

Fano Sequential Decoding Algorithm. There are three basic sequential decoding algorithms, the Wozencraft-Reiffen (Ref 27:154-156) (W-R), stack (Ref 27:163-167) and the Fano. The Fano algorithm was used for the link simulations implemented for this thesis. Of the three algorithms listed above, the Fano and stack algorithms are the most commonly used. The Fano algorithm is generally considered to be the most practical to implement in hardware (Ref 26:372), and commonly used by commercially available decoders. To date, many sequential decoder performance studies and computer simulations also use the Fano algorithm. The stack algorithm is, in general, only used in computer simulations. A discussion of the comparison between the Fano and stack algorithm is deferred to a later section.

The Fano algorithm is a refined version of the W-R algorithm. Of particular interest is the Fano algorithm's capability to allow a decoding decision to be delayed as far beyond the first constraint length as desired (within limits of time and decoder storage capabilities). That is, a decoding decision of the oldest bit can be made after it successfully processed some number of data bits typically equal to three to five times the convolutional code constraint length.

The general procedure of sequential decoding is as follows. The decoder attempts to recreate the correct transmitted sequence by fitting the received data sequence into a tree which models the encoder. This procedure is accomplished "sequentially" using appropriate v-tuples of received symbols. Each v-tuple will fit precisely into the tree only if no channel errors have occured. Otherwise, the decoder will have to select a trial path which does not correspond to the received sequence. The severity of the difference between the received sequence and trial path is calculated using a metric (to be described). Periodically, the metric for a trial path is compared with a running threshold. Under certain circumstances the trial path is revised and/or the threshold is revised.

The Fano algorithm will work with a wide variety of decoding metric functions. A commonly used metric for this sequential decoding technique is the logarithm of the likelihood ratio:

$$\log \frac{p(y|x)}{p(y)} \tag{32}$$

for the channel output y and input x, where y and x are taken to be branches of the code tree. Fano modified this form to include a bias term, giving the branch metric or likelihood function for the nth branch of the code tree:

$$\lambda_n = \log_2 \frac{p(y_n|x_n)}{p(y_n)} - B \tag{33}$$

Branch likelihood functions are combined to give a path likelihood function:

$$L_n = \sum_{i=1}^{n} \lambda_n \tag{34}$$

whose value and behavior for $n = 1, 2, \ldots$ determine the movements of the decoder through the code tree. This distance measure or path likelihood function, L_n, is a monotonically increasing function of the a posteriori probabilities $p(y_n|x_n)$. Assuming that all paths are equiprobable, the above Eqs (33) and (34) result (Ref 1:11). The input x_n is one of the 2^b v-tuples corresponding to the branch of the code tree stemming from node $n-1$. For the sequential decoder simulations performed in this analysis $b = 1$. The v-tuples correspond to the number of modulo-two adders implemented by the convolutional encoder. Therefore, x_n is one of two 2-tuples generated by using previous (tentative) digit decisions in a duplicate of the encoder, y_n is the corresponding received v-tuple, and the constant B is chosen such that L_n will increase on the average for correct path choices and decrease otherwise. Computer simulations by Jordan (Ref 20:283-297) indicate the best choice for the bias term is, $B = R$.

The running threshold levels T_j with which values of L_n are compared are taken to be uniformly spaced with increments ΔT. The parameter ΔT affects the amount of com-

putational effort during a search. As ΔT is reduced the decoder becomes more responsive to incorrect decisions but at the same time makes it more prone to label a correct path as incorrect due to noise. This results in numerous short searches. On the otherhand, a large value ΔT means a correct path is less likely to be confused with an incorrect one due to noise but the decoder will take longer to respond to a wrong decision. This results in fewer but longer searches. The best choice for ΔT depends on channel conditions (Ref 1: 13).

If binary data is input to the encoder, then there is a binary representation of the transmitted and trial sequences throughout the encoding and decoding process. The input to the decoder, however, doesn't have to be a binary representation, i.e., 1 or 0. In fact 2^n quantized levels (i.e., "soft" decisions with n-bit quantization) is often used as a method for preserving and utilizing additional information about the received channel noise. Note, $n = 1$ represents the case of "hard" decisions, i.e., either a 0 or 1 as the received channel digit. It should be noted, regardless of the number of quantization levels, the use of a binary code means that each node in the code tree can have exactly two branches stemming from it and leading to the node corresponding to the next information bit. The effect of the quantization, in essence, produces additional values for branch and path metrics.

To explain the Fano sequential decoder, it is helpful to use a concept of the search tree. A search tree corresponds to a code tree but has a different function. The search tree displays in quantitative terms the values of the various branch and path metrics which result as the decoder moves through the code tree along various trial sequences. Referring to the metric of Eq (33), upward motion in the search tree is related to an increase in L_n . An increasing L_n corresponds, in general, to correct hypotheses, regardless of whether a 1 or 0 was sent. For example, a bit with low level channel noise, which may have been correctly hypothesized will produce a branch metric value at some terminal node in the search tree that is higher than it would have been in the event of more intense received noise.

Before discussing the Fano sequential decoding algorithm, it is important to understand the concept of being at a node or looking forward or backward to a node. In each of these cases, the path metric value for the node in question is computed, and a comparison is made between this value and the current threshold. The distinction between the two concepts becomes apparent when the node which is serving as a reference point for the current portion of the search for the correct path is considered. Being at (or moving to) a node signifies that a decision has been made to accept tentatively the branch and trial sequence terminating at that node. In looking ahead (or back) to a node, there is only a temporary

look at the node, with acceptance only a possibility.

The Fano sequential decoding algorithm is shown in a general form in Figure 18. The flow diagram here is applicable to any desired decoding metric. Either the Hamming distance or a likelihood ratio can be used for the decoding metric. The quantized integer symbol metric performs the same task as the log likelihood functions. The Hamming distance and likelihood ratio metrics result, respectively, in decreasing and increasing trends in the value of the path metric for the correct path. On a short term basis (3-4 nodes), the path in the search tree may head in a direction opposite to that expected for the correct path. This is usually a result of increased channel noise. However, it is the long term development of the path metric and search tree path that is of most importance.

The portion of Figure 18 to the left of the dashed line constitutes the forward loop, in which the decoder advances through the search tree. Decoding decisions can be obtained for any digits (in the simulations performed $b = 1$) that are at least one constraint length behind the most recent accepted branch. Branch acceptance occurs in the "move ahead" box in the forward loop and when the threshold is satisfied. The Fano algorithm may produce a decision at this point but usually defers a decision until three to five constraint lengths have been successfully processed. Unlike the W-R algorithm, the Fano algorithm allows this strategy

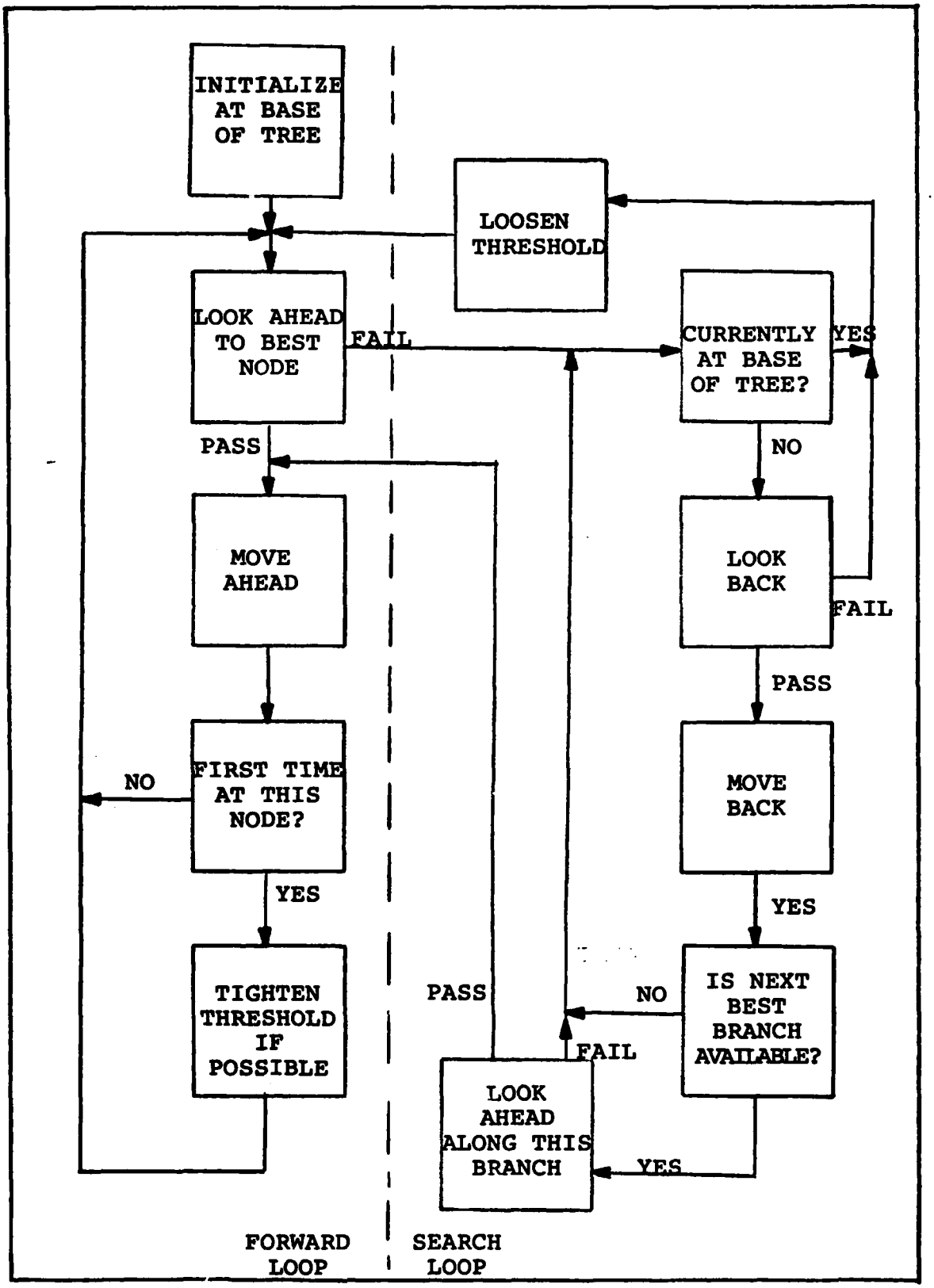


Figure 18. Fano Sequential Decoding Algorithm Flow Diagram.

to be used and thus, reduces the probability of a decoding error. This technique, however, requires additional storage for received digits and branch metrics plus increased bookkeeping.

The portion of Figure 18 to the right of the dashed line constitutes the search loop. A decoding computation is defined to take place whenever the decoding metric is computed for a branch. This occurs whenever the "look-ahead" box is executed.

The forward loop continues to be executed as long as the path metric resulting from adding the newest branch satisfies the current threshold. This results in the "pass" path from the "look-ahead to best node" box. If the new path metric value doesn't satisfy the current threshold a "fail" decision is made and the decoder goes into the search mode. Upon entry into the search loop, the first thing the decoder must do is to determine if it is already at the base of the tree. If it is, and since the best branch did not pass, the decoder cannot advance until the threshold is loosened. This step is repeated if necessary until the path metric corresponding to the best branch satisfies the tightened threshold.

If the decoder is not at the base of the tree then it must "look-back" to the preceding node in the path. Next, the path metric value at this node is tested to determine if it satisfies the current threshold. The threshold is loosened if the path metric value doesn't satisfy the threshold,

and the decoder returns to the forward loop to attempt to extend the path with this looser threshold, starting at the node from which it just looked back. However, if the value of the path metric at this node satisfies the current threshold, the decoder actually moves back to this node and begins to try other branches (next best branch) if one or more are available, the decoder returns to the forward loop when a branch is found which enables the path metric to pass. If, on the other hand, no satisfactory branch is found at this retreated node, the decoder, after checking as to whether it is at the base of the tree, backs up another node.

The operation of the Fano sequential decoding algorithm is best understood by working through an example. This example is focused on the Fano algorithm itself without going through the process of calculating branch and path metrics. A discussion of branch and path metric calculations will be covered in a later section.

The search tree in Figure 19 shows only the path metric values and thresholds. It is irrelevent for this example to know whether a branch at a node corresponds to a 0 or 1 information digit. The metric used in this example is any function (such as the logarithm of the likelihood ratio or sum of quantized integer symbol metrics) which tend to increase with sequence length. That is, the overall trend is upward but there may be short term decreases in the metric value.

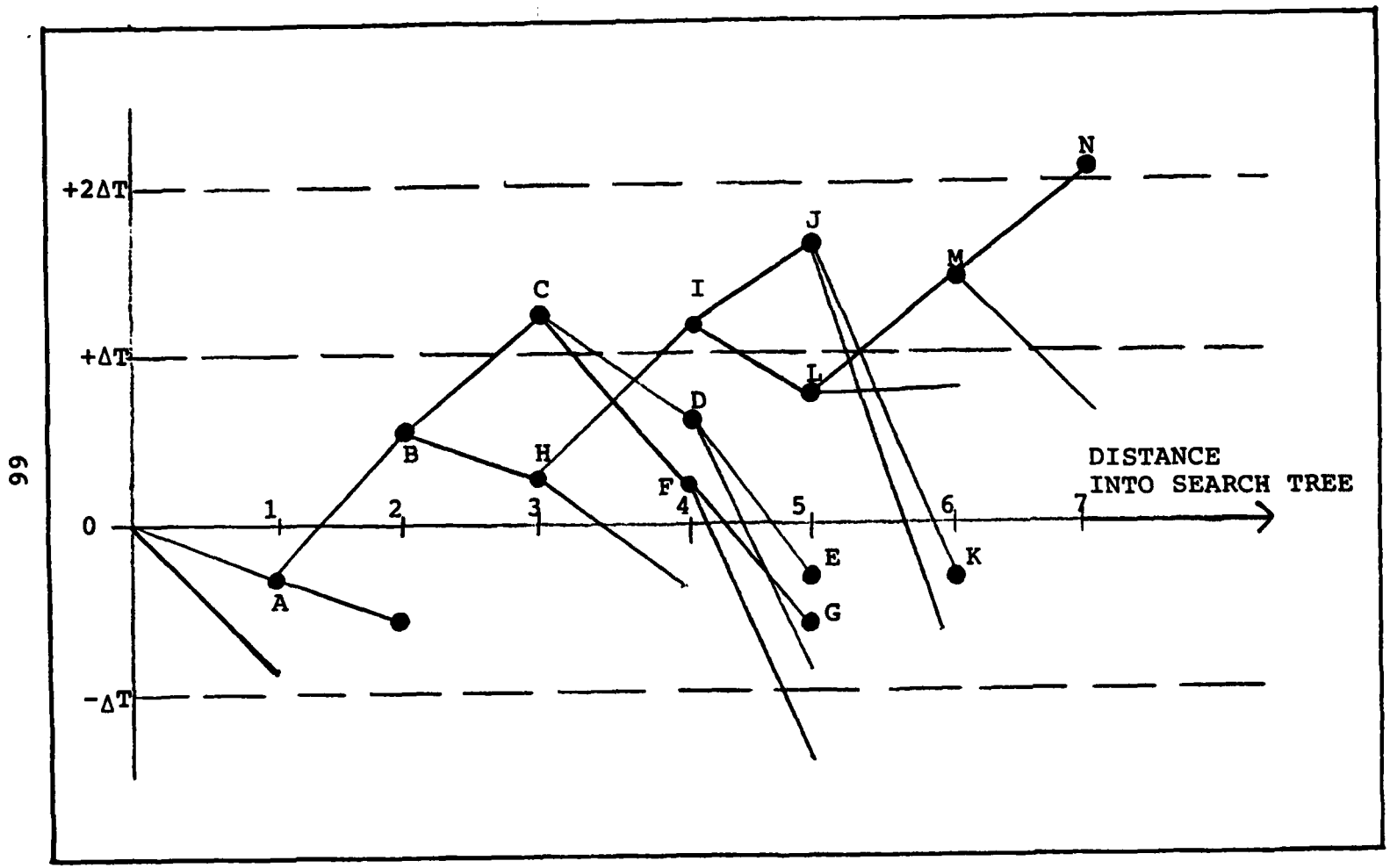


Figure 19. Search Tree Paths for the Likelihood Function Decoding Metric.

TABLE II

Operation of the Fano Algorithm

STEP	POINTER LOCATION	THRESHOLD VALUE	ACTION		
1	ORIGIN	0	TEST A FAIL SET T=- T		
2	ORIGIN	$-\Delta T$	TEST A PASS MOVE TO A		
3	A	$-\Delta T$	TEST B PASS MOVE TO B		SET T=0
4	B	0	TEST C PASS MOVE TO C		SET T=$+\Delta T$
5	C	$+\Delta T$	TEST D FAIL TEST B	FAIL	SET T=0
6	C	0	TEST D PASS MOVE TO D		
7	D	0	TEST E FAIL TEST C	PASS	MOVE TO C
8	C	0	TEST F PASS MOVE TO F		
9	F	0	TEST G FAIL TEST C	PASS	MOVE TO C
10	C	0	TEST B PASS MOVE TO B		
11	B	0	TEST H PASS MOVE TO H		
12	H	0	TEST I PASS MOVE TO I		SET T=$+\Delta T$
13	I	$+\Delta T$	TEST J PASS MOVE TO J		
14	J	$+\Delta T$	TEST K FAIL TEST I	PASS	MOVE TO I
15	I	$+\Delta T$	TEST L FAIL TEST H	FAIL	SET T=0
16	I	0	TEST J PASS MOVE TO J		
17	J	0	TEST K FAIL TEST I	PASS	MOVE TO I
18	I	0	TEST L PASS MOVE TO L		
19	L	0	TEST M PASS MOVE TO M		SET T=$+\Delta T$
20	M	$+\Delta T$	TEST N PASS MOVE TO N		

In Figure 19, the abscissa shows path length starting at the base; the ordinate specifies the value of the path metric and shows the various possible threshold values. A qualitative understanding of the Fano algorithm operation can be obtained by correlating each step of Figure 19 and Table II with movement through the generalized flow diagram of Figure 18.

One important point needs to be introduced in regard to this example that isn't clearly evident from the flow diagram of Figure 18. In step 16 of the example, the threshold is not increased in conjunction with the tentative, passing, forward movement of the search tree to the node at J . If the threshold was allowed to increase at this point in the example the decoder would get into an infinite search loop and thus never process any further into the code tree. Therefore, a controlling parameter is required to prevent the decoder from getting into this situation. The Fano algorithm utilizes a flag (Fano flag) which is a binary variable to prevent or allow the threshold to increase. For example, the Fano flag is set to 1 when the algorithm enters the search loop and set to 0 in the forward loop mode. A more detailed explanation can be found in Fano's paper (Ref 9:72) and the following section on decoder implementation.

Before leaving the Fano algorithm presentation, a brief discussion of the Hamming metric and how it is used may prove helpful for demonstrating the difference between the various

decoder metrics and resultant search tree path. The branch metric using the Hamming distance decoding metric is given by

$$\lambda_n = d_H - B \tag{35}$$

where d_H is the Hamming distance between the trial information sequence and the corrupted received sequence, and B is the bias term. The path metric, L_n, is again the sum of the branch metrics, λ_n, for all the branches of the tentatively accepted search path:

$$L_n = \sum_{i=1}^{n} \lambda_n \tag{34}$$

As previously mentioned, the overall trend of the search path will be decreasing when the Hamming metric is used. Figure 19 has been redrawn in Figure 20 to demonstrate this trend.

Variability of Decoder Computations. A major problem with sequential decoding is the great variability, as a function of time, in the number of decoder computations necessary to extend the trial sequence by one branch. Thus a similar variability exists which related to the number of computations required per information digit decoded. This variability has a significant effect on decoder hardware for real time operation or on cost of computer time for off-line operation. A sequential decoder using the Fano algorithm will remain in the search loop for long periods of time when a particularly noisy received sequence is encountered. These

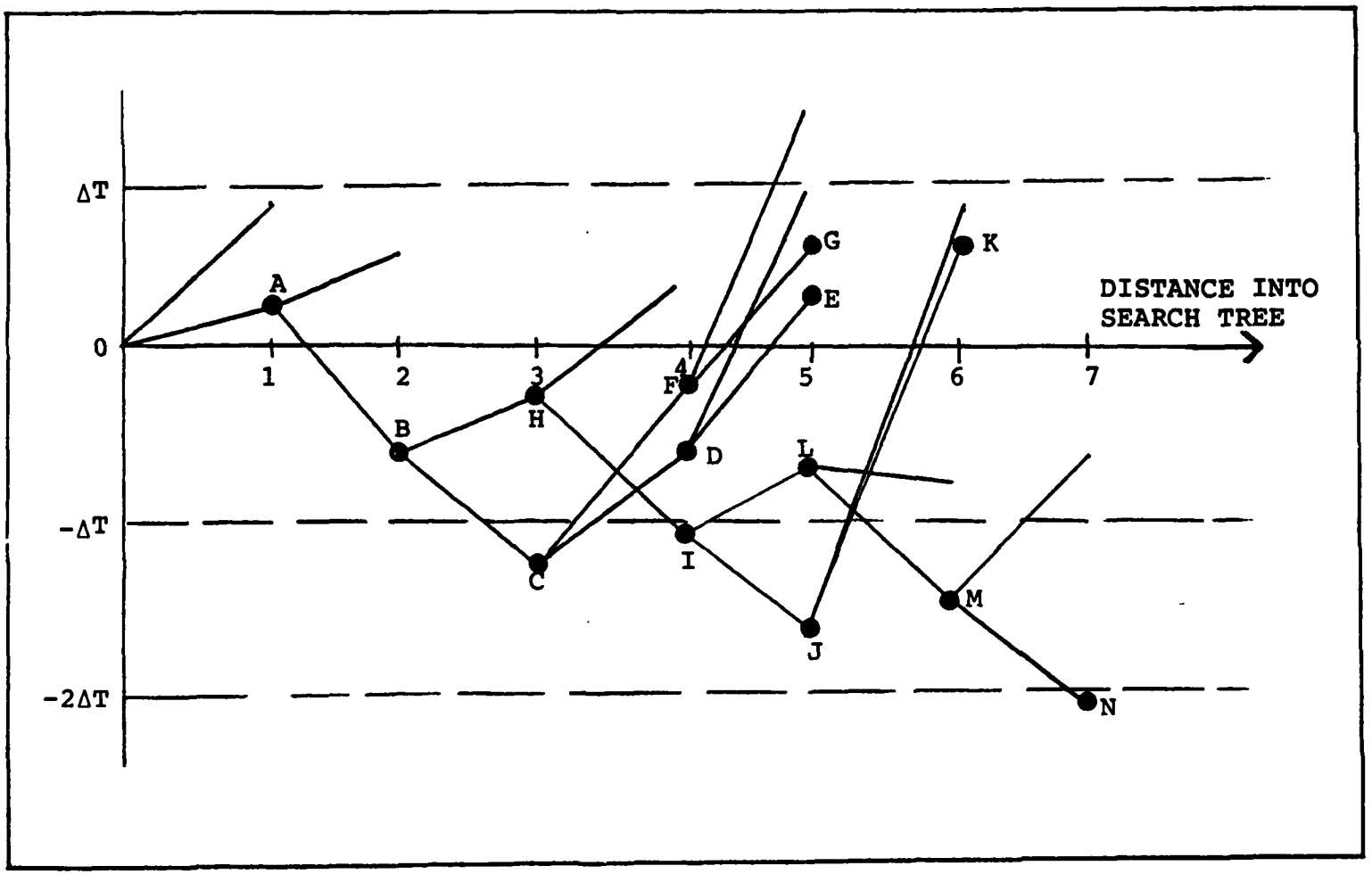


Figure 20. Paths in Search Tree for the Hamming Distance Decoding Metric.

long searches necessitate accumulation of the continuing stream of in-coming digits. In order to preclude the loss of these digits an input buffer must be available. If, however, the decoder tries to search a node for which received data has passed out of buffer memory, an overflow is said to occur. Thus to minimize the number of digits which enter the buffer during a search, the decoder must have a speed advantage on the order of ten times the information rate (Ref 27: 168). The decoder computational effort depends strongly on the ratio R/R_{comp} . Other techniques for handling the overflow problem are given by Odenwalder (Ref 23:163-164). For sequential decoding applied to a DMC, Gallagher (Ref 14: 279) provided a bound for the average number of forward moves (in the search tree) per decoded subblock of $k = vR$ information digits:

$$4/[1 - e^{v(R-R_{comp})}]^2 \quad , \quad R < R_{comp} \qquad (36)$$

where v = number of channel digits per subblock, R = code rate in bits per channel symbol or $R = k/v$. The important point is the above expression becomes infinite as $R \to R_{comp}$. In general, the ratio R/R_{comp} should be less than 0.09 .

Summary of Sequential Decoding Utility. Sequential decoding is a powerful technique for convolutional codes because it can provide bit error rates which decrease exponentially with code constraint length at information rates below

channel capacity and necessarily so below R_{comp} . The undetected error probability with sequential decoding can be made as small as desired by increasing the code constraint length. Because decoder complexity is only a weak function of the constraint length, long constraint length codes are practical for sequential decoding (Ref 23:164-166). Finally, because of the limitations due to the variability of computational time to advance one node or branch, sequential decoding is unsuitable as a burst-correcting technique without extensive data bit and/or symbol interleaving.

Sequential Decoding Algorithm Selection

An extensive selection process for a sequential decoding algorithm took place during the early stages of this thesis. The possibilities were narrowed down almost immediately to the stack and Fano algorithms. Although the stack algorithm has the same performance as the Fano and requires a reduced number of computations per decoded bit, it was not selected in favor over the Fano technique. The modest increase in number of branch computations of the Fano algorithm is more than offset by its advantage of requiring a considerable reduction of decoder processing storage requirements compared to the stack (Ref 26:373). Computer memory is the trade-off that must be made for the speed advantages of the stack decoder. In fact, because of the memory requirements of the stack algorithm, it is primarily implemented only on the

computer. As a result of this latter additional fact and evidence which shows the Fano algorithm to still be the most practical to implement, the Fano decoder was selected. A general discussion of the stack algorithm is provided by Wiggert (Ref 27:163-167).

Design Requirements

The primary purpose behind the development of the sequential decoder subroutine was to design a subprogram that would easily integrate into the current existing AFWL FSK-PSK Code. In general, the sequential decoder subroutine had to function with the AFWL Code in much the same way the existing Viterbi decoder subroutine functions. That is, the sequential decoder subroutine had to be not only a digital simulation model of such a device but in fact a sequential decoder implemented in a Fortran-coded computer program. In addition, the sequential decoder subroutine and main AFWL PSK program had been designed in such a way to enable the sequential decoder only when the user desired the decoder to be implemented in a particular link simulation. Finally, the AFWL sponsor specified the sequential decoder should be able to decode, at least, rate 1/2 convolutional codes with constraint lengths 25 through 30.

The resultant sequential decoder subroutine design, which will be presented in detail at the end of this chapter, more than adequately satisfies the above design requirements.

Specifically, the decoder can handle convolutional codes up to constraint length 32 and binary code rates 1/2, 1/3, 1/4, 1/5, 1/6, 1/7 and 1/8. In addition, the sequential decoder can perform 1, 2 or 3-bit soft decision decoding and provide an estimate of the total number of bit computations in any particular simulation. The latter information can provide the communication systems analysis with a starting point in determining the potential overflow problems that may occur when an existing off-the-shelf sequential decoder is implemented in a 2400 bps, DBPSK satellite link. In short, the AFWL FSK-PSK Code sequential decoder subroutine provides a viable baseline representation of the performance one can expect to see in the AWGN and nuclear scintillated channel for an EHF, 2400 bps, DBPSK, convolutionally encoded, and sequentially decoded link.

Code Symbol Quantization for Soft Decision Sequential Decoding

The purpose of this section is to describe the soft decision decoding process and why it was used in the 2400 bps DBPSK link simulations for both the AWGN and nuclear scintillated environment. The DBPSK demodulator can provide either hard decision or soft decision outputs for use in the sequential decoding process, or for that matter, any error-correction decoding process. Hard decisions use a single binary digit (0 or 1) to represent each demodulated bit, with the binary decisions made on the basis of the outputs

of the integrate-and-dump matched filters in the demodulator data channels at the end of each bit period. Soft decisions come about by quantizing the matched filter outputs to more than two levels, thus, requiring more than one binary digit to represent each demodulated symbol bit. As a result, the additional quantization levels provide the decoder with a measure of the quality of each received code symbol that must be processed. The sequential decoder uses this higher quality received information to improve its efficiency in tracing a path through the code tree which better represents the actual transmitted information sequence. In theory, the greater the number of quantization levels, the better the performance of soft decision sequential decoding. However, it has been found (Ref 19:495-496) that a smaller number of quantization levels provide near-optimum decoder performance. In fact, most soft decision decoding are 8-level (3-bit) code symbol quantization. Figures 21 and 22 from Odenwalder (Ref 23:44, 51) demonstrate the advantages of soft decisions over hard decisions and the small coding gain of infinitely fine quantization over 8-level quantization.

Figure 21 shows the E_b/N_o ratio required to operate at the computational cutoff rate, R_{comp} , for a coherent BPSK modulated AWGN channel with 1, 2, 3 and infinite bits of quantization of the demodulator outputs versus the code bandwidth expansion. The code bandwidth expansion is simply one over the code rate, R . Note how 3-bit soft quantization

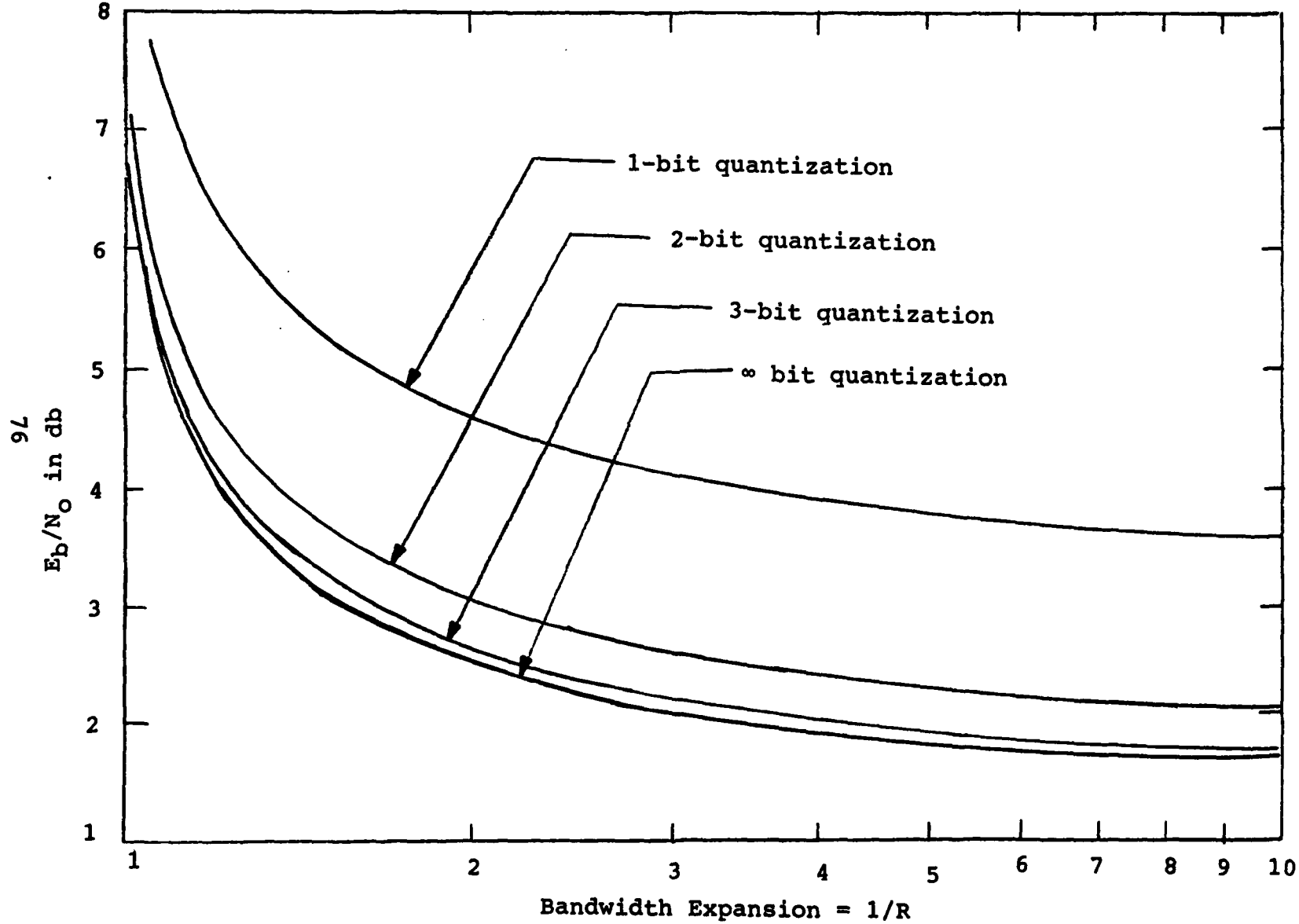


Figure 21. E_b/N_o Required to Operate at $R=R_{comp}$ for a Coherent BPSK AWGN Channel (Ref 23:44).

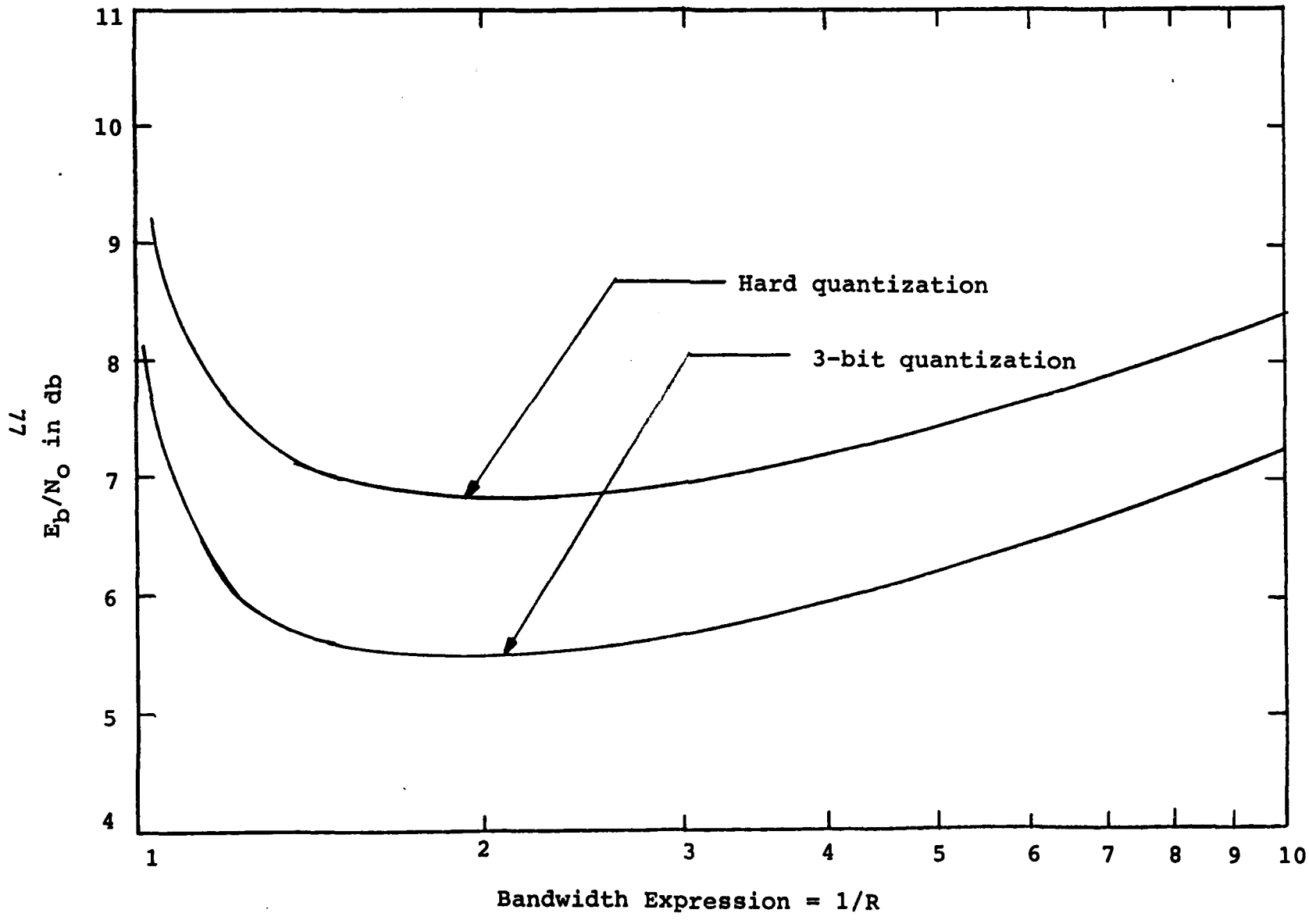


Figure 22. E_b/N_o Required to Operate at $R=R_{comp}$ for an Interleaved DBPSK Channel (Ref 23:51).

is almost equivalent to infinitely fine quantization. Figure 22 shows the E_b/N_o ratio required to operate at $R = R_{comp}$ versus the bandwidth expansion for 1- and 3-bit DBPSK demodulator quantization. This figure demonstrates the gain of 3-bit soft decisions over hard decisions for a rate 1/2 code is approximately 1.3 dB. Because of the near-optimum decoder performance of 8-level quantization this scheme was used in all of the soft decision sequential decoding link simulations.

The soft decision quantization thresholds in the demodulators are taken at specified fractions of the mean value of the logic '1' and logic '0' levels at the outputs of the integrate-and-dump matched filters. An automatic gain control, AGC, is used to hold the signal level near some design value. These design values are the mean values of the logic '1' and '0' levels set at plus and minus a known design constant (Ref 3:3-123).

The number of quantization levels, Q , is typically some power of two ($Q = 2, 4, 8$, etc.). As described by Bogusch, the designer of this section of the AFWL FSK-PSK Code, the thresholds for the demodulated code symbol quantization are set at $2L/Q$ times the design data level, where $L = 0, 1, 2, \ldots, (Q/2) - 1$. For hard decisions, $Q = 2$, there is only one threshold at zero. For $Q = 4$ there are three thresholds at 0 and 1/2 of the mean value. For 3-bit quantization, $Q = 8$, there are seven thresholds at 0, 1/4, 1/2, and 3/4 of the mean level.

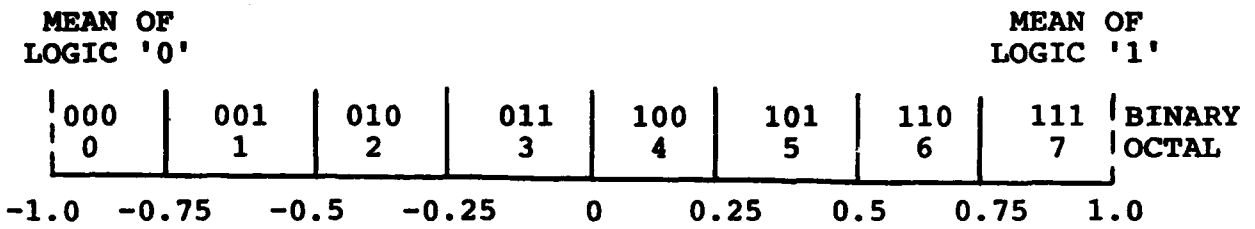


Figure 23. Demodulator Thresholds for 8-Level Soft Decision Code Symbol Quantization.

An illustration of the 8-level quantization is shown in Figure 23. Note the corresponding binary and octal representation of the quantized code symbols in each uniformly spaced voltage interval. The convention used here is a logic '0' corresponds to a negative voltage (π phase shift), and a logic '1' corresponds to a positive voltage (no phase shift). It is interesting to note that the most significant bit in each quantization interval is just the hard binary decision that would result from using a single threshold at zero to determine the polarity of the demodulator output. The two most significant bits are those that would result from 4-level quantization of the output.

As previously mentioned, the quantized code symbols provide a measure of the reliability of each binary decision. Using Figure 23, a binary number 000 (or integer symbol 0) represents, by adopted convention, a highly reliable binary

TABLE III

Integer Symbol Metric Table for Rate 1/2 Encoded Link and 8-Level (3-bit) Quantization

Additive White Gaussian Noise Channel

Code Symbol	Received Quantized Symbol 0	1	2	3	4	5	6	7
0	4	4	2	0	-8	-20	-34	-58
1	-58	-34	-20	-8	0	2	4	4

zero and the numbers 001 (1), 010 (2), etc., indicate received zeros with decreasing reliability. Similarly, the binary number 111 (or integer 7) represents a higher reliable received binary one, and the numbers 110 (6), 101 (5), 100 (4), etc., indicate received ones with decreasing reliability. For example, a value of 4 is interpreted as slightly more likely to be a binary one than a binary zero, but the quality of the decision is considered to be poor. The soft decision sequential decoding algorithm implemented in these simulations assigns weights to each input code symbol based on the quantized numerical value provided by the demodulator. These weights are called integer symbol metrics.

Table III lists the integer symbol metrics used by the sequential decoding algorithm for a rate, $R = 1/2$, convolutional code and 8-level (3-bit) quantization. These integer symbol metrics are the same decoding metrics used by Costello (Ref 15:416) in his work with the Fano decoder for

for the AWGN channel. The metrics correspond to Fano's expression for the likelihood function decoding metric,

$$\lambda_n = \log\frac{p(y_n|x_n)}{p(y_n)} - B$$

in Eq (33), scaled and rounded to integer values. Branch metrics are computed by adding the integer symbol metrics for the two received code symbols on the branch. This is clearly illustrated in the following section on the sequential decoder subroutine design.

Recall from the section on the Fano sequential decoding algorithm, integer symbol metrics like the ones in Table III, can be used to perform the same function as the logarithm of a likelihood ratio decoding metric. It was suggested both decoding metrics generated a path metric value whose overall trend was upward. The concept of the integer symbol metric can best be demonstrated using an approach provided by Viterbi and Omura for the BSC (Ref 26:350-354) with hard decisions.

Let the probability of transition, $p = p(y_n=1|x_n=0)$ or $p(y_n=0|x_n=1)$ be equal to 0.03, $p(y_n) = 1/2$ for the BSC, and the bias term $B = R = 1/2$. Using Eq (33), likelihood ratio decoding metric values can be represented by

$$\lambda_n = \log_2[2(1-p)] - B \qquad (37)$$
$$= 0.456$$

TABLE IV

Integer Symbol Metric Table for a BSC Channel

Binary Symmetric Channel			
p	Code Symbol	Received Symbol 0	1
0.03	0	2	-20
	1	-20	2

where

$$x_n = y_n \quad ,$$

and

$$\lambda_n = \log_2[2p] - B \qquad (38)$$
$$= -4.56$$

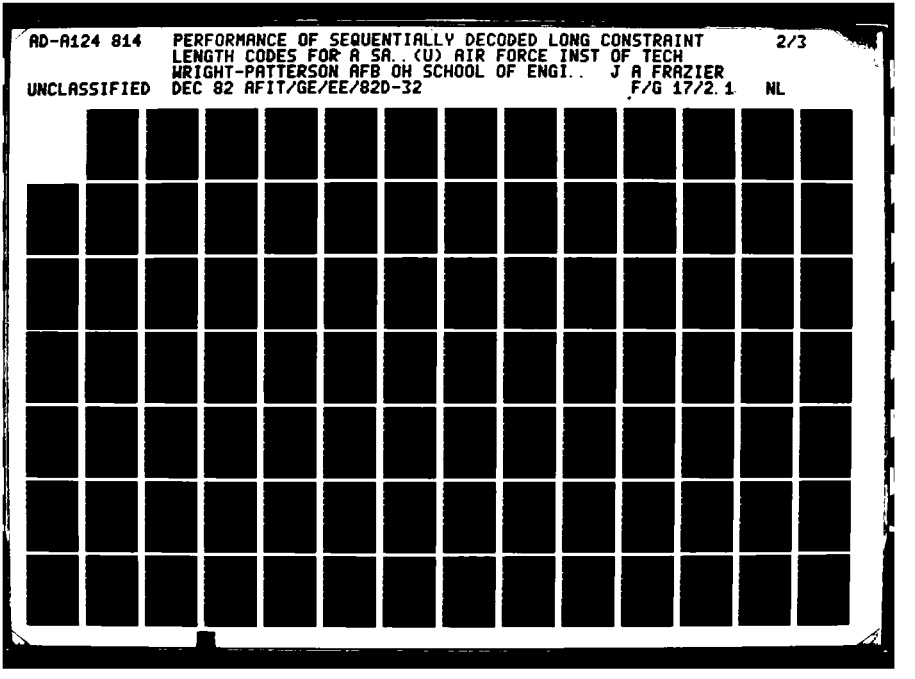
where

$$x_n \neq y_n \quad .$$

The integer symbol metrics for this BSC are actually derived from the likelihood ratio metric values by multiplying by some positive constant and scaling to some desired integer symbol metric value:

$$\lambda_n = \begin{matrix} .456\ (1/.456) = +1 \\ -4.56\ (1/.456) = -10 \end{matrix}$$

and scaled by 2 gives the integer symbol metric values for the BSC shown in Table IV.


AD-A124 814
PERFORMANCE OF SEQUENTIALLY DECODED LONG CONSTRAINT LENGTH CODES FOR A SA.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J A FRAZIER
2/3
UNCLASSIFIED
DEC 82 AFIT/GE/EE/82D-32
F/G 17/2.1
NL

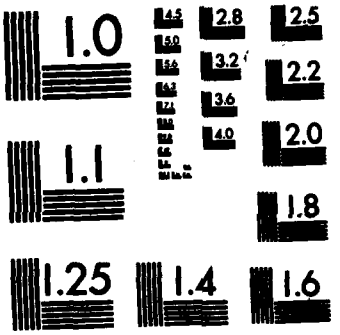


TABLE V

Integer Symbol Metric Tables for 1 and 2-Bit Soft Decision Decoding

Threshold Increment (T) = 4
Hard Decisions (2 level quantization)

Code Symbol	Received Quantized Symbols 0	1
0	0	-8
1	-8	0

Threshold Increment T = 8
2-Bit Soft Decisions (4 level quantization)

Code Symbol	Received Quantized Symbols 0	1	2	3
0	2	0	-8	-20
1	-20	-8	0	2

The current sequential decoder subroutine is designed to implement 3-bit soft decisions with the integer symbol metrics of Table III for rate 1/2 convolutional codes, and a threshold increment of $\Delta T = 32$. Although not tested, Table V, suggests integer symbol metrics and corresponding threshold increments that may work for 1 and 2-bit soft decision decoding with rate 1/2 convolutional codes for the AWGN channel. Utilizing either of these sets of integer symbol metrics in place of the 3-bit soft integer symbol metrics, by the sequential decoder, requires minor modifications to the subroutine.

The integer symbol metrics and threshold increments corresponding to 1, 2, and 3-bit quantization for convolutional code rates 1/3 through 1/8 require further investigation.

Sequential Decoder Implementation

Decoder Input Mode Test. Each time the sequential decoder is called by the main program, the decoder is examined to determine if this is an initial entry into the subroutine. Upon initial entry, the program proceeds to the initial entry section of the decoder; otherwise, control is transfered to the input section.

Initial Entry Section. The first function that occurs in this section is a quantitative comparison between the decoder path memory length and the convolutional code constraint length to enable the sequential decoder. If this requirement is satisfied the decoder starts initializing parameters.

Several parameters required by the Fano sequential decoding algorithm are initialized to zero. These include the data bit counter, threshold, Fano flag, origin node path metric value, node counter and the search tree stack. The search tree stack stores the quantitative values of the various branch and path metrics which result as the decoder advances to those nodes of the code tree that make up the search tree paths.

Other parameters initialized in this section include the received symbol counter, duplicate encoder state, number

of encoder states, and the threshold increment. The threshold increment, $\Delta T = 32.0$, used in this subroutine is currently designed for a receiver utilizing 3-bit quantization integer symbol metrics and a rate 1/2 convolutional code. This value was suggested by Geist (Ref 15:416) as a result of Fano decoder performance analyses for the additive white Gaussian noise (AWGN) channel.

The initial entry module also initializes logic switches which control the decoder input and output modes. The input mode switch is turned on to enable decoder control to jump directly to the input section upon future accesses to the subroutine during a particular simulation. The decoder output mode indicator performs an important function of signaling whether or not data bit decisions are being made from the decoder. Its logical value is changed when the decoder has successfully processed enough bits, equal to the decoder path memory length, to enable the decoder to output the oldest bit in the search tree path. The decoder path memory length is typically 4 to 5 times the code constraint length. Both the decoder path memory length and code constraint length are user specified parameters.

Finally, the significance of the parameters discussed so far will become more evident later in this algorithm presentation. Before leaving this section, note that all the arithmetic parameters are integers except for the threshold value and threshold increment.

Input Section. The input section is where the sequential decoder operation actually begins. In this section, the main fuction is to store the quantized received code symbols, corresponding to the current nth order node in the code tree, in the stack. The number of symbols stored in the stack corresponds to the number of modulo-two adders (i.e., the binary code rate) of the convolutional encoder implemented in this simulation. Therefore, this leads to the secondary purpose of the input section which is to keep track of the number of code symbols it is receiving from the demodulator or deinterleaver. That is, code symbols are input to the decoder from the deinterleaver if interleaving is implemented in the simulation. The demodulator and deinterleaver subroutines of the AFWL FSK-PSK Code actually output their code symbols to the main program, and then the decoder subroutine receives the code symbols, one at a time, from the main program upon each call to that subroutine. Because one code symbol is received at a time, the input section of the decoder works to obtain a full symbol set before it allows decoder precessing to continue.

The sequential decoder subroutine will return to the main program as many times as necessary until a full symbol set is received. The maximum number of code symbols that can be received by this decoder is eight, corresponding to a rate 1/8 convolutional encoder. When a full symbol set is obtained, the symbol counter is reset to zero.

Duplicate Convolutional Encoder Section. Duplicating the convolutional encoder at the channel input is necessary in the implementation of the sequential decoder using the Fano algorithm. The duplicate encoder's design in the decoder subroutine is patterned after the one utilized in the convolutional encoder section in the main program of the AFWL FSK-PSK Code.

The purpose of the duplicate convolutional encoder is to generate the two possible sets of code symbols (channel inputs) and two possible encoder states that could have been produced by the convolutional encoder in the main program (data channel encoder). The encoder state and corresponding encoder output code symbol set is associated with either a '0' input data bit or a '1' input data bit, refer to the chapter on convolutional encoding. Recall, the primary function of the sequential decoder is to produce a path within the binary code tree (search tree path) which represents the most likely transmitted channel code symbols and encoder state transitions. The duplicate encoder enables the sequential decoder to store only two possible encoder states and channel input sets that could have been produced and select among these the most probable for extending the search tree path. Otherwise, the sequential decoder would have the massive task of generating and storing all of the encoder states and channel code symbols that make up the code tree, and selecting among all of these the most probable for extending

the search tree path. For example, a digital communication system utilizing a rate 1/2, constraint length, $K = 25$, convolutional encoder would require the decoder to deal with $2^{(25-1)} = 16,777,216$ states.

Because this sequential decoder implementation works only with binary codes, the duplicate convolutional encoder generates the possible channel input symbols and encoder states for the branches of the code tree corresponding to a '0' input data and a '1' input data bit. It is important to note upon initial entry into the encoder section the current encoder state is initialized to zero. Both the encoder state for the '1' input data bit branch and the encoder state for the '0' input data bit branch are labeled by the decoder for future manipulation in a follow-on section of the decoder. In this follow-on section, the decoder determines that encoder state which corresponds to the best and worse branch respectively, and updates the current duplicate encoder state.

The encoder output symbols are stored in an (2 X 8) dimensional integer array. An example of this array structure is shown in Figure 24. As shown by this figure the rows correspond to the two possible binary bit values of each branch emanating from a particular node in the code tree, and the number of columns which have the possible encoder output code symbols (possible channel input symbols) correspond to the number of modulo-two adders implemented by the duplicate convolutional encoder. These encoded symbols for each branch

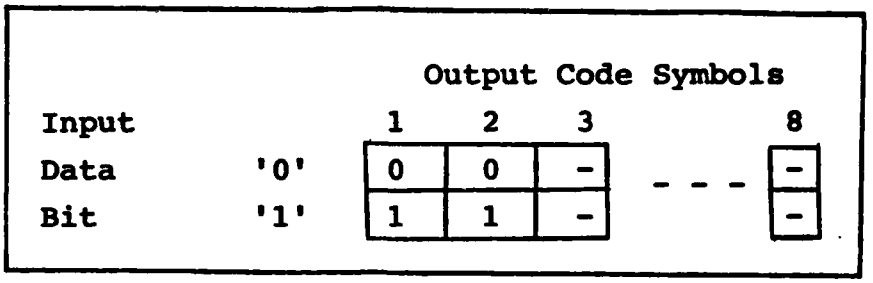


Figure 24. Example of a Encoder Output Code Symbol Array for a Rate 1/2 Code.

are eventually used to calculate the branch metric values in the code tree. Finally, the duplicate encoder section functions only when the decoder is ready to attempt to advance to the next node in the code tree.

Search Tree Stack. Before moving onto a discussion of the follow-on sections of the decoder, it may prove helpful to explain the concepts and structure surrounding the sequential decoder's search tree stack. The search tree stack is an important part of the sequential decoder subroutine. The description will aid in understanding the branch metric, path metric, and encoder state manipulations that will take place in the sequential decoder. Figure 25 and Table VI are referenced throughout the discussion of this section.

The search tree stack is an integer array for storing branch and path metric values, encoder state values, receiver code symbols and two binary flag bits. One of the flags represents the data bit corresponding to the current tentatively accepted branch. The other flag indicates whether the tentatively accepted branch corresponds to the best or worst

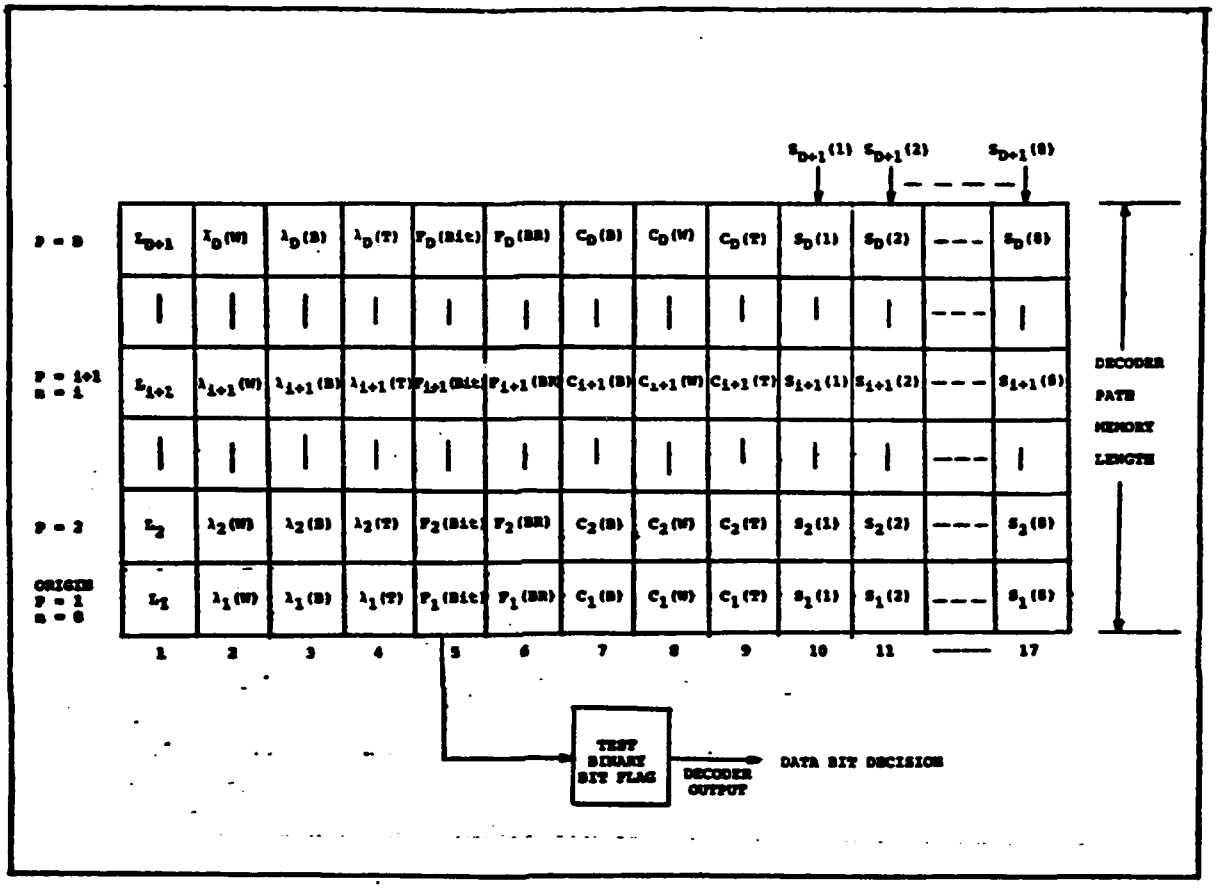


Figure 25. Search Tree Stack.

TABLE VI

Search Tree Table

L_{n+1}	= current accepted path metric
$\lambda_{n+1}(W)$	= worst branch metric
$\lambda_{n+1}(B)$	= best branch metric
$\lambda_{n+1}(T)$	= tentatively accepted branch
$F_{n+1}(Bit)$	= data bit corresponding to $\lambda_{n+1}(T)$
$F_{n+1}(BR)$	= best or worst branch flag corresponding to $\lambda_{n+1}(T)$
$C_{n+1}(B)$	= encoder state corresponding to $\lambda_{n+1}(B)$
$C_{n+1}(W)$	= encoder state corresponding to $\lambda_{n+1}(W)$
$C_{n+1}(T)$	= encoder state corresponding to $\lambda_{n+1}(T)$
$S_{n+1}(K)$	= received code symbol, 8 maximum corresponding to Rate 1/8 code
P	= stack pointer
n	= node number
D	= decoder path memory length

branch at some node. It is called a search tree stack for the following reason. In Fano sequential decoding, it is helpful to make use of the concept of a search tree, which corresponds to (but is not the same as) the code tree. The search tree displays quantitatively, in terms of values of the various branch and path metrics, and encoder states which results; the movement of the decoder through the code tree along various trial sequences. The stack implemented here is used to store these branch and path metric values along with the encoder states associated with the advancing nodes.

The branch metrics are carried along for the purpose of recalculating the path metrics. An example of this requirement occurs when the current accepted branch metric value for a particular node is changed from the best branch metric value to the worst branch metric value. A path metric recalculation is also done for the situation where the search tree digresses back to some previously examined node. Such a procedure occures in the search loop of the Fano algorithm.

The encoder state values carried along in the stack are those values corresponding to the best and worst branches for a particular node in the code tree. In addition, the current encoder state is carried along as well; corresponding to the current accepted encoder state at some specific nodal distance in the search tree. When the current tentatively accepted metric value is changed from the best to worst or, vise versa,

the current encoder state is changed to reflect the same best to worst or worst to best alterations. Therefore, the current encoder state, upon which the search tree stack will extend, is always associated with the correct node when forward movement into the code tree is attempted. Finally, the received code symbols corresponding to a specific nodal distance in the code tree, are used for calculating the branch metric values at some node.

Another important reason for the sequential decoder search tree stack entails an advantage of using the Fano algorithm. The Fano algorithm can produce a decoding decision anytime the trial sequence has reached a length of one constraint length and has satisfied the threshold. One advantage of the Fano algorithm is that it can produce a decision at this point but generally is programmed to defer a decision until three to five constraint lengths have been successfully processed. This strategy reduces the probability of a decoding error.

Upon initial entry into the sequential decoder routine, the stack stores all required branch and path metric, encoder state and received code symbol values until the decoder has successfully processed a number of data bits equal to the decoder path memory length. The decoder path memory length is typically three to five times the convolutional encoder constraint length. The integer array which makes up the stack

is completely filled when the last row equal to the decoder path memory length is filled, refer to Figure 25.

After the stack is completely filled the oldest bit can be outputted. In fact, the data bit output logic switch which was turned off is now turned on and data bits are outputted from that point forward throughout the rest of the simulation. Everytime a data bit is outputted or a decoding decision is made the stack is reset. That is, all branch and path metric, encoder state and received code symbol values at the stack origin disappear and the values in the row above move down to take their place. This is accompanied by a digression of all the rows one space down leaving the last row empty.

A stack pointer is utilized to indicate which row will be written into or read out of in the Fano sequential decoding process. After the stack has been filled once, the new branch and path metric, encoder state and received code symbol values associated with the next advancing node in the code tree will be stored in the last row.

The branch metrics include the following: best branch metric, worst branch metric and the tentatively accepted branch metric. The best branch metric, $\lambda_{n+1}(B)$, is associated with the greater of the two branches emanating from a node of order n . That is, it is the '0' branch metric or the '1' branch metric, which ever is greater. Following, the worst branch metric, $\lambda_{n+1}(W)$, corresponds to the lesser of

the two metric values.

When a node is addressed for the first time, the tentatively accepted branch metric, $\lambda_{n+1}(T)$, corresponds to $\lambda_{n+1}(B)$. In addition, the two binary flags are set to the tentatively accepted branch. It is quite possible, however, the value of $\lambda_{n+1}(T)$ could become $\lambda_{n+1}(W)$ in the Fano sequential decoding process. The path metric value is represented by L_{n+1} .

The encoder states include the best encoder state, worst encoder state and the current encoder state. The best encoder state value, $C_{n+1}(B)$, corresponds to the best branch, and the worst branch. The current encoder state, $C_{n+1}(T)$, corresponds to the current tentatively accepted branch.

A decoding decision is made by examining the data bit flag, $F_{n+1}(Bit)$. $F_{n+1}(Bit)$ takes on a binary value, '0' or '1'. If $F_{n+1}(Bit)$ equals '0', a '0' data bit decision is made. Similarly, when $F_{n+1}(Bit)$ equals '1', a '1' data bit decision is made.

Calculation of Branch Metrics. The major function of this important section of the sequential decoder subroutine is to calculate the branch metrics on the '0' branch and '1' branch of the code tree. Before this phase of the process can occur, however, another function of storing the integer symbol metrics is performed.

Each integer symbol metric is quantized to a specified value (0→1, 0→3, or 0→7, depending on simulation input

parameters found in the PSK data input file). The integer symbol metrics are generated and stored in a (2 X 8) array in accordance to the two possible binary data symbols and the quantized integer symbol as received from the demodulator or deinterleaver. The integer symbol metric values are assigned for 3-bit quantization using a table like the one shown below.

TABLE VII

Integer Code Symbol Metrics for 8-Level (3-bit) Receiver Quantization

		Received Quantized Code Symbols							
		0	1	2	3	4	5	6	7
Data Symbols	'0'	4	4	2	0	-8	-20	-34	-58
	'1'	-58	-34	-20	-8	0	2	4	4

Consider this 8-level receiver quantization example. If the soft-quantized received code symbol was 0, the integer symbol metric value in the '0' row of the array would be 4. This is because a received code symbol of 0 represents, by adopted convention, a highly reliable received binary code symbol '0', and integer values of 1, 2, on up to 7, indicate received '0's with decreasing reliability. Of course, a soft-quantized received code symbol of 0 then forces the integer symbol metric value in the binary code symbol '1' row of the array to take on a value of -58. If, for example, the quantized received code symbols 1 and 5 were received for a rate

1/2 channel encoded satellite communication link, the integer symbol metric array would be as shown in Figure 26. Note, the row number 1 represents the binary code symbol '0' and the row number 2 represents the binary code symbol '1'. This is necessary for integer symbol metric array addressing and manipulation.

		Quantized Received Code Symbols	
		1	5
Data Symbols	'0' → 1	4	-20
	'1' → 2	-34	2

Figure 26. Example of a Decoder Integer Symbol Metric Array (IQUANT) for a Rate 1/2 Convolutional Encoded Link.

It is evident from Figure 26 that the two most likely demodulated binary code symbols were '0' and '1'. The full integer metric set is stored in the decoder and used later on to calculate decoder branch metrics at a particular node in the code tree.

Branch metrics are calculated by summing individual quantized integer symbol metrics. Using the previous examples of the integer symbol metrics, Figure 26, and the branch code symbols, Figure 24, the branch metric for the '0' branch would be,

$$\lambda_{n+1}(0) = \text{IQUANT}\ (1,\ 1) + \text{IQUANT}\ (1,\ 2)$$
$$= 4 + (-20)$$
$$= -16$$

because the code symbols on the '0' branch were 00. The branch metric for the '1' branch would be,

$$\lambda_{n+1}(1) = \text{IQUANT}(2,1) + \text{IQUANT}(2,2)$$
$$= -34 + 2$$
$$= -32$$

because the code symbols on the '1' branch were 11. If on the otherhand, the code symbols on the '0' branch were ⊥0 the branch metric for that branch would be,

$$\lambda_{n+1}(0) = \text{IQUANT}(2, 1) + \text{IQUANT}(1, 2)$$
$$= -34 + (-20)$$
$$= -54$$

The next major function is relating the best or most likely branch and the worst branch from the '0' and '1' branch metrics that were just calculated. The best branch is then tentatively selected as the current branch upon which the search tree path will be extended. If the '0' branch metric is equal to the '1' branch metric, the '0' branch is arbitrarily selected as the best branch and thus the tentatively accepted branch for extending the search tree path. The best, worst, and current accepted branch metric values are stored in the search tree stack as $\lambda_i(W)$, $\lambda_i(B)$, and $\lambda_i(T)$ respectively, refer to Figure 25.

In addition to the branch metric calculations and manipulations that take place in this section, we now go back and

consider the '0' branch and '1' branch encoder states that were generated by the duplicate convolutional encoder. The encoder states corresponding to the best branch and worst branch are selected and loaded into the search tree stack, refer to Figure 25. In a similar manipulation that took place with the best branch metric value, the encoder state corresponding to the best branch is set as the current state upon which the search tree path will be extended. This function is accompanied by updating the current encoder section.

One last major function takes place in the best branch metric calculation section. The data bit flag in the stack is set to a value, '0' or '1', corresponding to the tentatively accepted branch (best branch), and the branch flag is set to 0 corresponding to the best branch.

Path Metric Calculation. When the branch metric values are calculated and stored in the stack, a path metric calculation can take place. The path metric is derived by summing together the path metric value from the previous node and the tentatively accepted branch metric value at the current node of interest. Referring to the notation used in the stack discussion, $L_{n+1} = L_n + \lambda_{n+1}(T)$.

Upon entry into the path metric calculation section of the subroutine, one of three different kinds of path metric calculations can take place. The current output mode of the sequential decoder plays a major role in the selection of which path metric is to be calculated. For example, if the

stack has been filled once and data bit decisions are being made by the decoder, the decoder checks to see if the path metric is to be calculated at the stack origin. The path metric calculation here involves summing together the tentatively accepted branch metric value at the stack origin and a value corresponding to the previous path metric stored at the stack origin before the entire stack was reset as done in the forward loop section of the Fano algorithm. Recall from the stack discussion, after a data bit decision is made, all of the old branch and path metric values disappear from the origin and new values from above move down and take their place. In order to calculate this path metric of interest, the old path metric value is retained. The second kind of path metric calculation that can take place, whether or not the logical output switch is on or off, is the normal path metric calculation which takes place beyond the stack origin.

A third potential path metric calculation entails that calculation performed to extend just beyond origin node $(n = 0)$. That is, the path metric value at the second node $(n = 1)$ of the code tree, refer to Figure 27. With the path metric value at the first node assumed to be 0, the path metric at the second node is equal to the tentatively accepted branch metric value at the second node, i.e.,

$$L_{n+1} = L_n + \lambda_{n+1}(T) \tag{39}$$

$$\begin{aligned} L_1 &= L_0 + \lambda_1(T) \\ &= 0 + \lambda_1(T) \\ &= \lambda_1(T) \end{aligned}$$

Fano Algorithm. The presentation here describes, in detail, the Fano algorithm implementation in the sequential decoding subroutine. The first function in the Fano algorithm is to determine if processeing will continue in the forward loop or search loop. This involves a comparison between the path metric value at the node of order (n+1) with the current threshold value. If the path metric is less than the threshold, decoder processing is transferred to the search loop; otherwise, processing moves to the forward loop.

Fano Algorithm Forward Loop Section. The discussion of this section will be aided by referencing Figure 28a. After decoder processing passes to the forward loop, an attempt is made to tighten the threshold. If the path metric is greater than the next higher incremented threshold value, the current threshold value is set (tightened) to that value. The node counter and stack pointer are then incremented by one.

The next step is to determine if the decoder is ready to receive the next set of code symbols from the main program. A return to the main program for the next set of code symbols only occurs if the data bit counter value is equal to the current advancing node counter value. If this latter condition is not satisfied, the sequential decoder returns to the duplicate encoder section and eventually tries again to move forward into the code loop. If the condition is satisfied, however, a return to the main program doesn't immediately occur . Several important steps must follow.

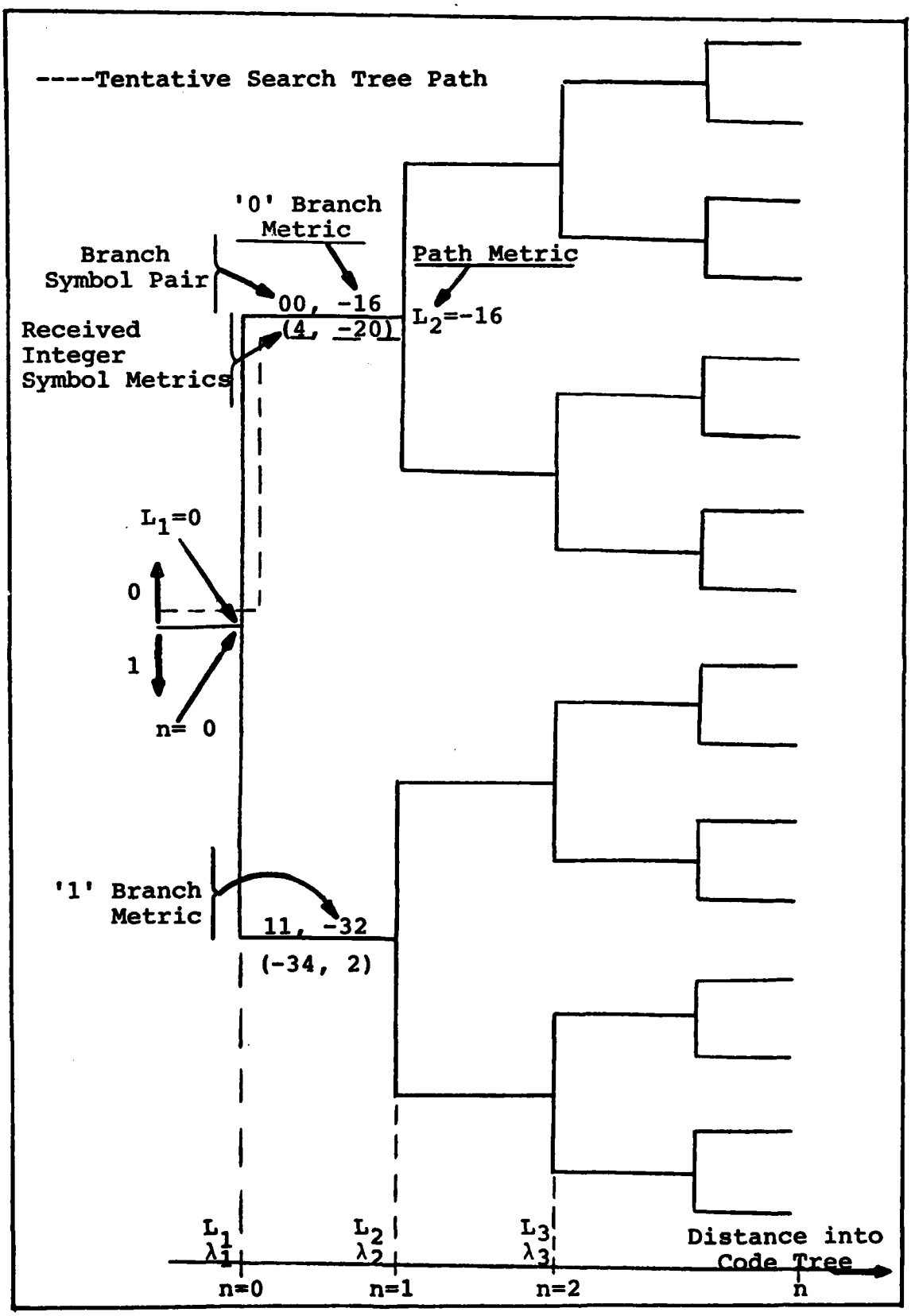


Figure 27. Code Tree Diagram using Integer Symbol Metrics.

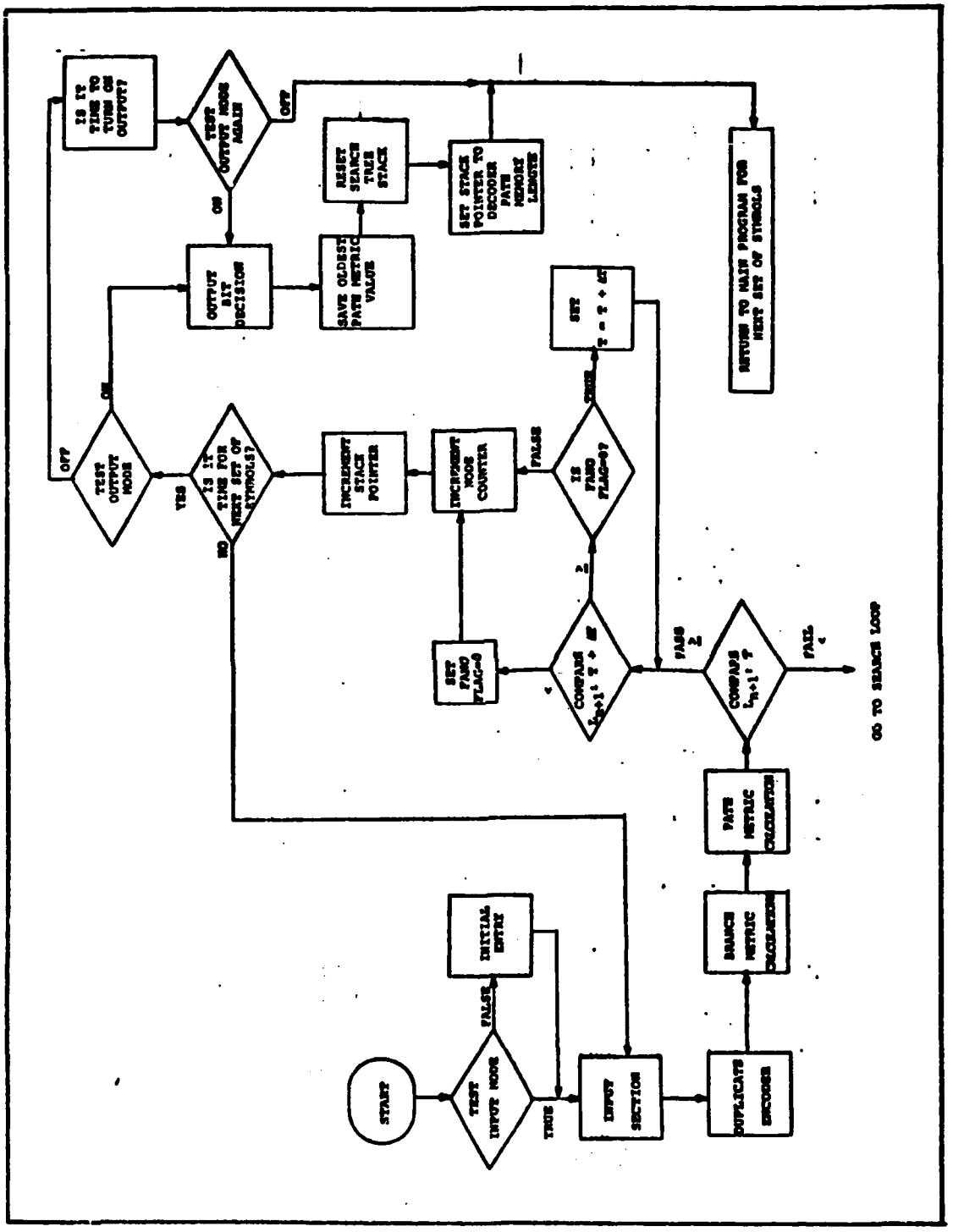


Figure 28a. Flow Diagram for Forward Loop of the Fano Algorithm.

First, the output mode is tested. If the current output mode indicator shows data bit decisions are not being made by the decoder, a test is performed to determine if the output should be turned on. The test involves performing a logical AND between the output mode and the data bit counter; and then comparing the result with the decoder path memory length. This test, in effect, checks to see if the search tree stack, with row dimension equal to the decoder path memory length, is completely filled for the first time. A test failure implies decoding decisions are not ready to begin; thus, program control returns to the main program to simulate the generation of the next data bit and eventually returns to the input section of the sequential decoder with a new set of demodulated or deinterleaved encoded symbols.

However, if the output mode indicates decoding decisions are being made, or the test for turning the output on passed, an output data bit decision is made in the manner presented in the section on the search tree stack. Following the decoding decision, the oldest path metric value located at the stack origin is saved, and the search tree stack is reset as previously discussed in the stack section.

Lastly, the stack pointer value is set to the decoder path memory length value because it is here, the last row in the stack, all newly calculated branch and path metric values; encoder state values, and received code symbols will be loaded. The stack pointer cannot advance any further

than the maximum row dimension of the search tree stack. Next, the sequential decoder subroutine transfers control to the main program to be called again and accept a new set of received code symbols.

Fano Algorithm Search Loop Section. Before discussing the search loop it is important to have a general understanding of the purpose for the Fano flag. A detailed discussion on the function of the Fano flag is presented in Fano's paper (Ref 9:72). In short, the Fano flag is a binary variable used to control a gate that allows or prevents the threshold from increasing. The choice depends on whether the Fano flag, F , equals 0 or 1. In the search loop, if all attempts fail to find a previous node that satisfies the current threshold, the threshold is lowered by ΔT and the decoder reverts to the first node which failed and tries to move forward. The threshold is fixed at this lower value $(T - \Delta T)$ until it finds an acceptable node which has not been reached before. In essence, the Fano flag holds the threshold value constant, and thus, prevents the algorithm from getting into a loop.

The sequential decoder search loop procedure will be described using Figure 28b. Some of the major and not so obvious algorithm functions will be presented.

When the current path metric of interest fails the Fano algorithm test discussed above, the decoder enters the search loop. After the Fano flag is set to one and the bit

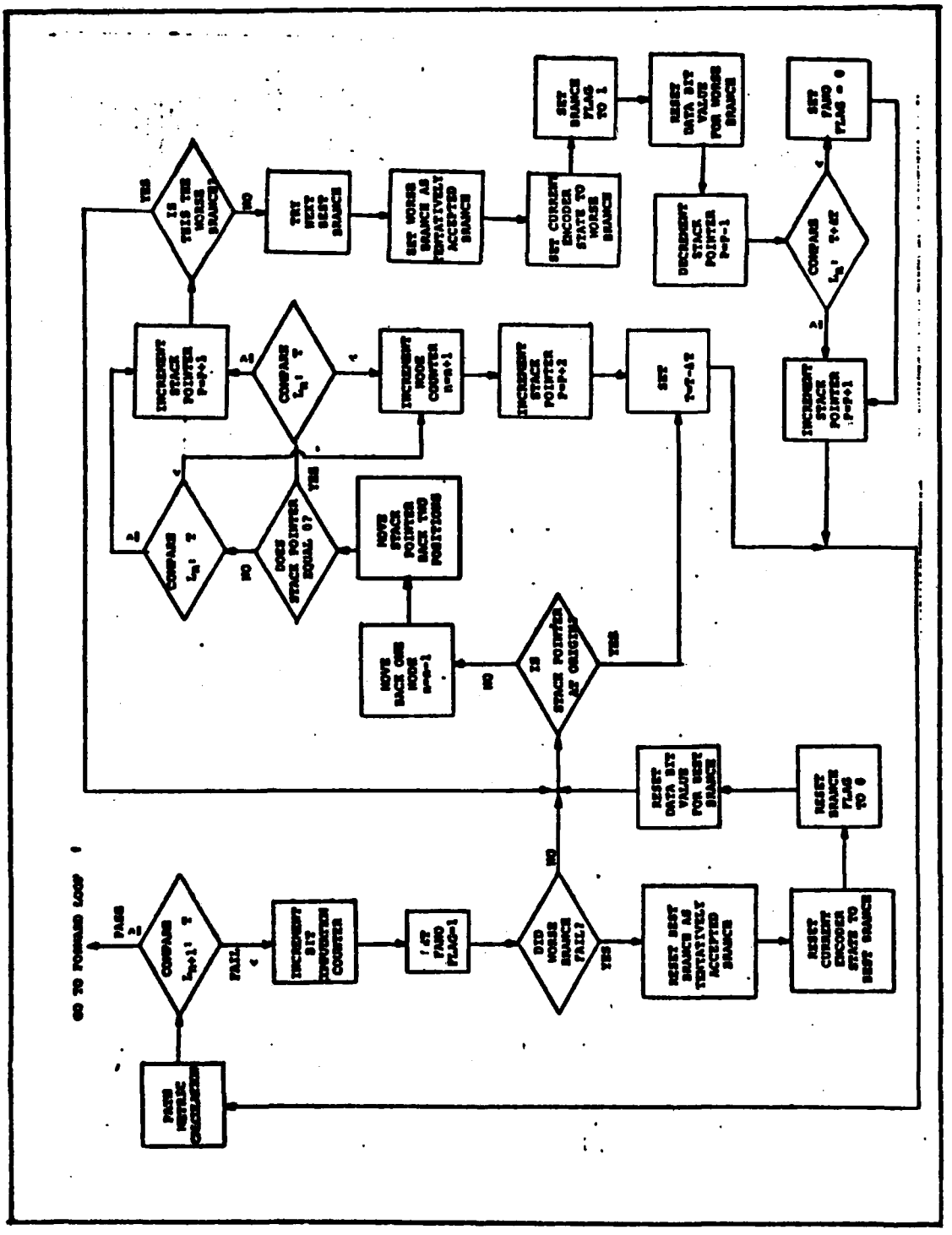


Figure 28b. Flow Diagram for Search Loop of the Fano Algorithm.

computation counter incremented by one, a test is performed to determine if the path metric which failed the prior threshold test corresponded to the worse branch extending from the current node of order n .

If so, the stack's tentatively accepted branch metric value and encoder state which currently correspond to this worse branch must be reset to correspond to the best branch. This task is followed by updating the current encoder state for later attempts at extending the search tree path, refer to Figure 28b. In addition, the branch flag is reset to zero (best branch) and the data bit flag value is changed using a logical EXCLUSIVE-OR operation.

Whether the worst branch or best branch failed, the next major step entails checking the stack pointer to determine if it is pointing to the origin stack position $(n = 0)$. If this is the case, the threshold is decreased by ΔT and decoder processing transfers back to the path metric section to attempt a forward movement into the code tree. However, if the decoder is not at the origin node, the stack pointer moves back two positions to enable a test of the path metric value one node back $(n = n - 1)$. Because of the way the stack pointer system is set up, the stack pointer value could be equal to zero after being decremented by two. In this situation, the path metric value to be tested in the following step will be that value at the code trec's origin $(L_n = L_0 = 0)$, if the stack hasn't been filled yet, or the

oldest path metric value that was retained in the forward loop section before the stack was reset.

If the path metric, one node back, is less than the current threshold (test failure), the node counter and stack pointer are re-incremented to their previous values before the latter test; the threshold is decreased by ΔT ; and the decoder moves back to the path metric section to attempt forward movement once again. When the path metric passes the latter test, the stack pointer is incremented by one, and the current tentatively accepted branch metric value located in that stack position is examined to determine if the least likely (worst) branch had been tested. If the least likely branch has been tested, the decoder returns to the origin test and search loop processing continues. On the otherhand, if only the best branch was tested, the decoder will try to extend the search tree path using the worst branch.

Trying the next best branch, first entails setting the tentatively accepted branch metric value and encoder state at the current stack position to correspond to this worst branch. This step is accompanied by updating the current encoder state for possible future attempts of extending the search tree path from this node. In addition, the data bit flag value and branch flag value are changed to correspond to the worst branch as well. Before the worst branch path metric is calculated and evaluated in the Fano algorithm test, the stack is decremented by one for the purpose of determining

if the path metric at this position will cause the Fano flag to remain at one or be reset to zero. This step is then followed by re-incrementing the stack pointer by one. If the worst path metric passes the Fano algorithm test, the search tree path attempts forward movement from that branch. However, if the worst branch path metric fails, search loop processing continues. Additional flow diagrams corresponding to the major sequential decoding functions which occur at the front end of the code are illustrated in Figures 29 through 33.

Before leaving this section of the sequential decoder's design, it is important to mention the fact that the number of bit computations, N , through the search loop is accumulated. The purpose of this function is explained in the next section.

Approximation of Time Spent in the Search Loop

The search loop section of the Fano algorithm includes an accumulator for storing the total number of bit computations through the search loop for any given link simulation. This bit computation value was used in the performance assessment tasks of this thesis to determine the average number of bit computations, N , required to decode a single bit. The value N provides a rough measure of the amount of time spent in a search. It is simply derived by dividing the accumulated bit computation value by the total number of decoded bits processed in any given link simulation, i.e.,

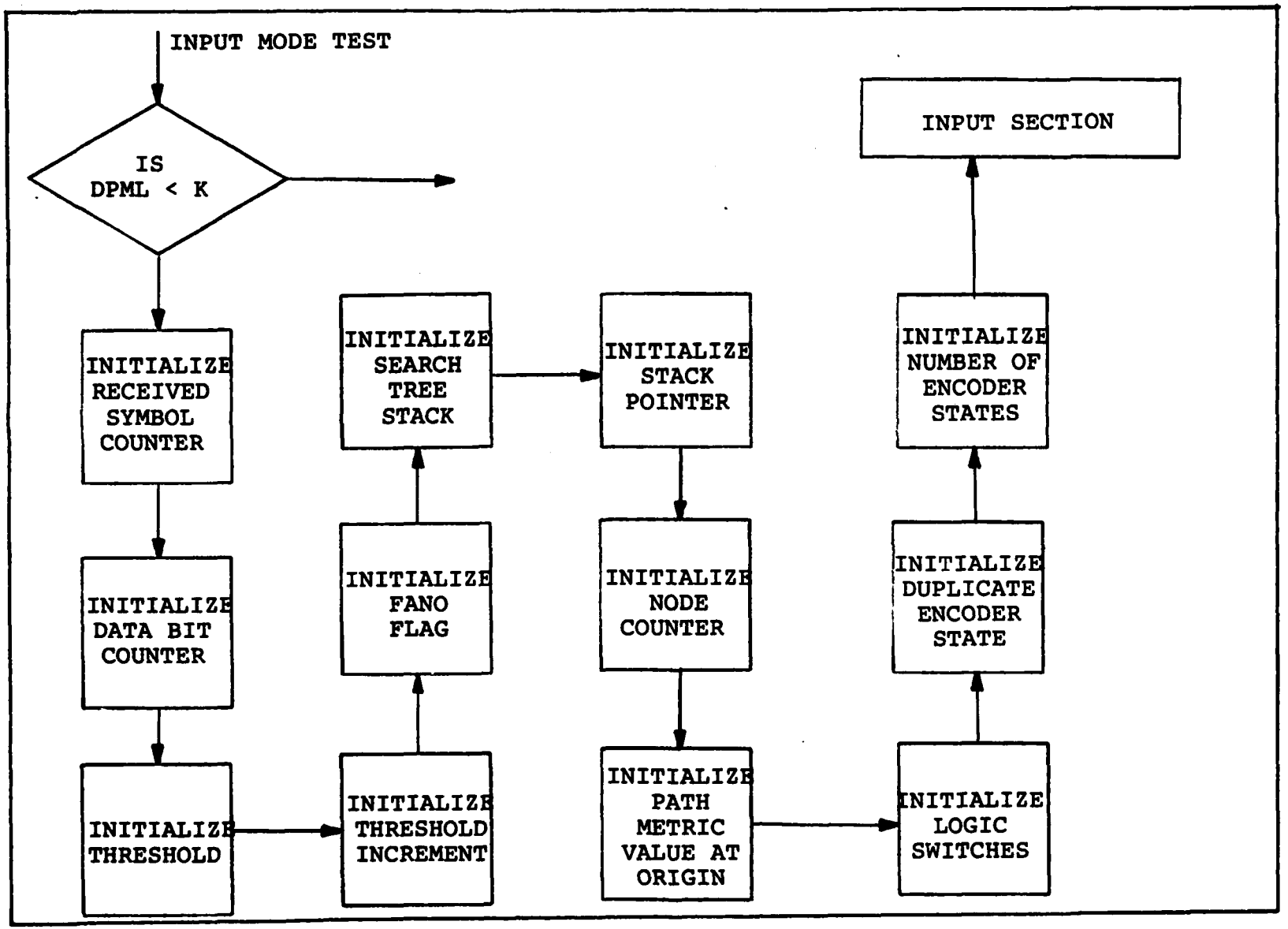


Figure 29. Flow Diagram of the Initial Entry Section.

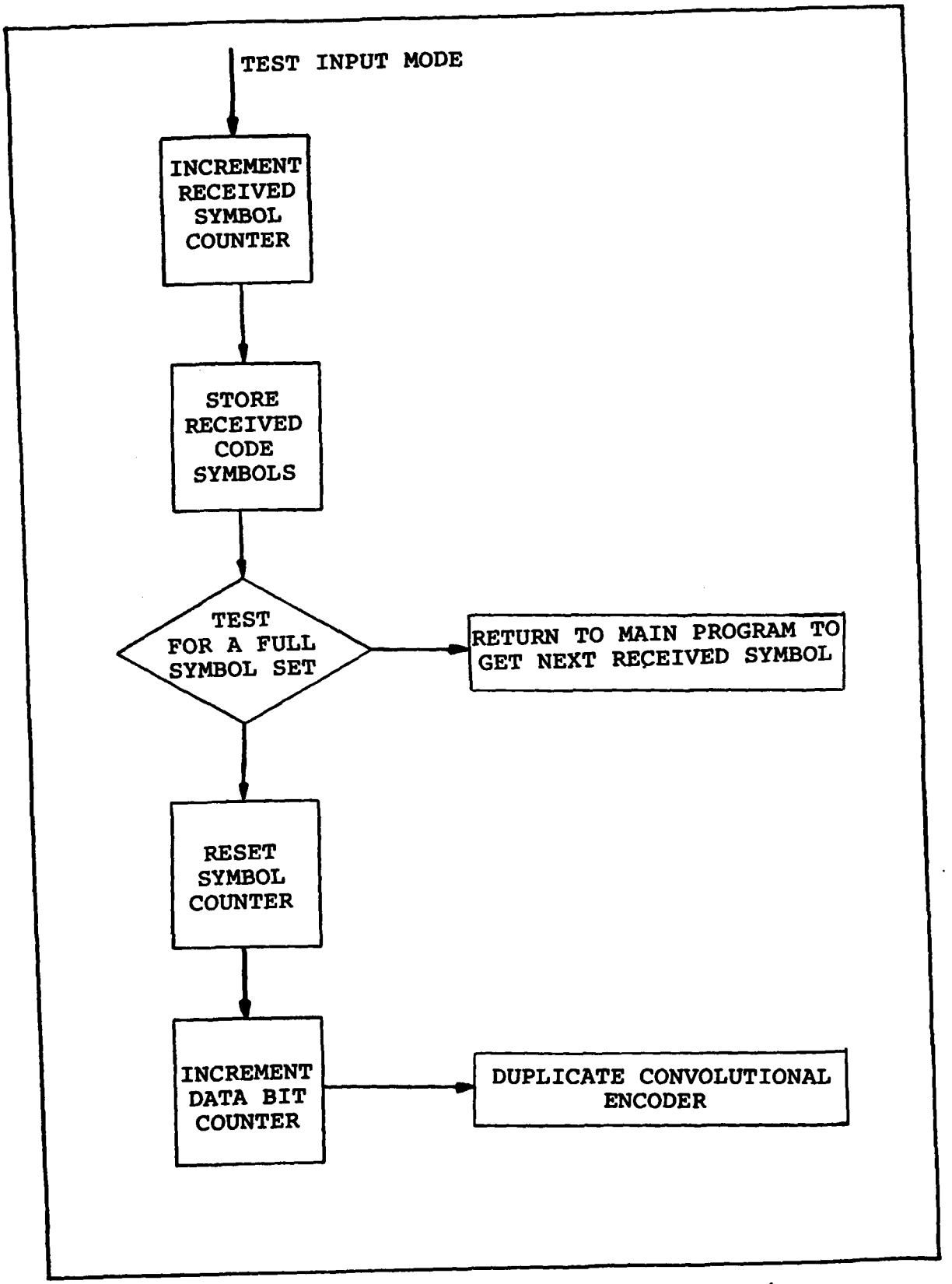


Figure 30. Flow Diagram of the Input Section.

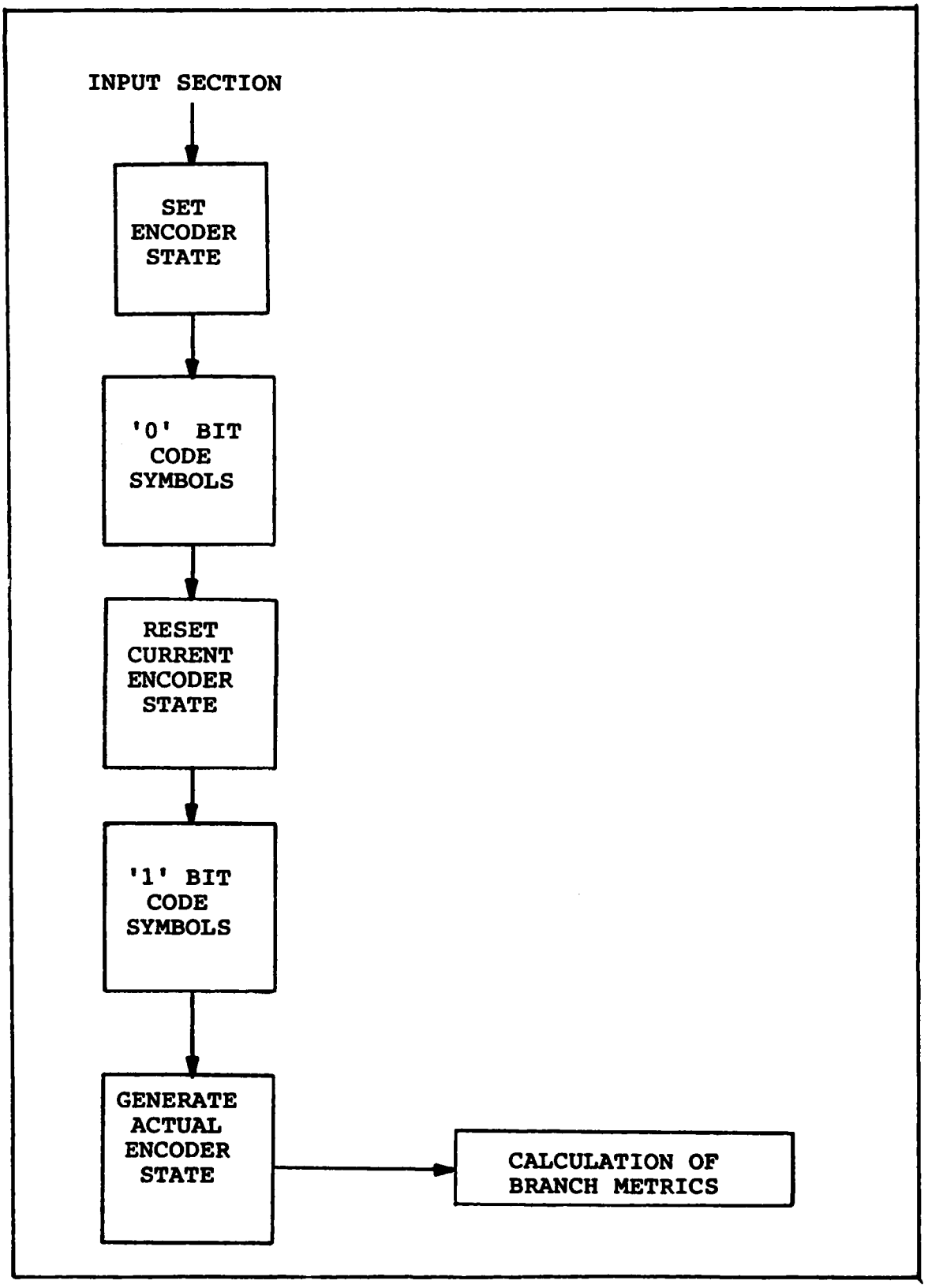


Figure 31. Flow Diagram of the Duplicate Convolutional Encoder.

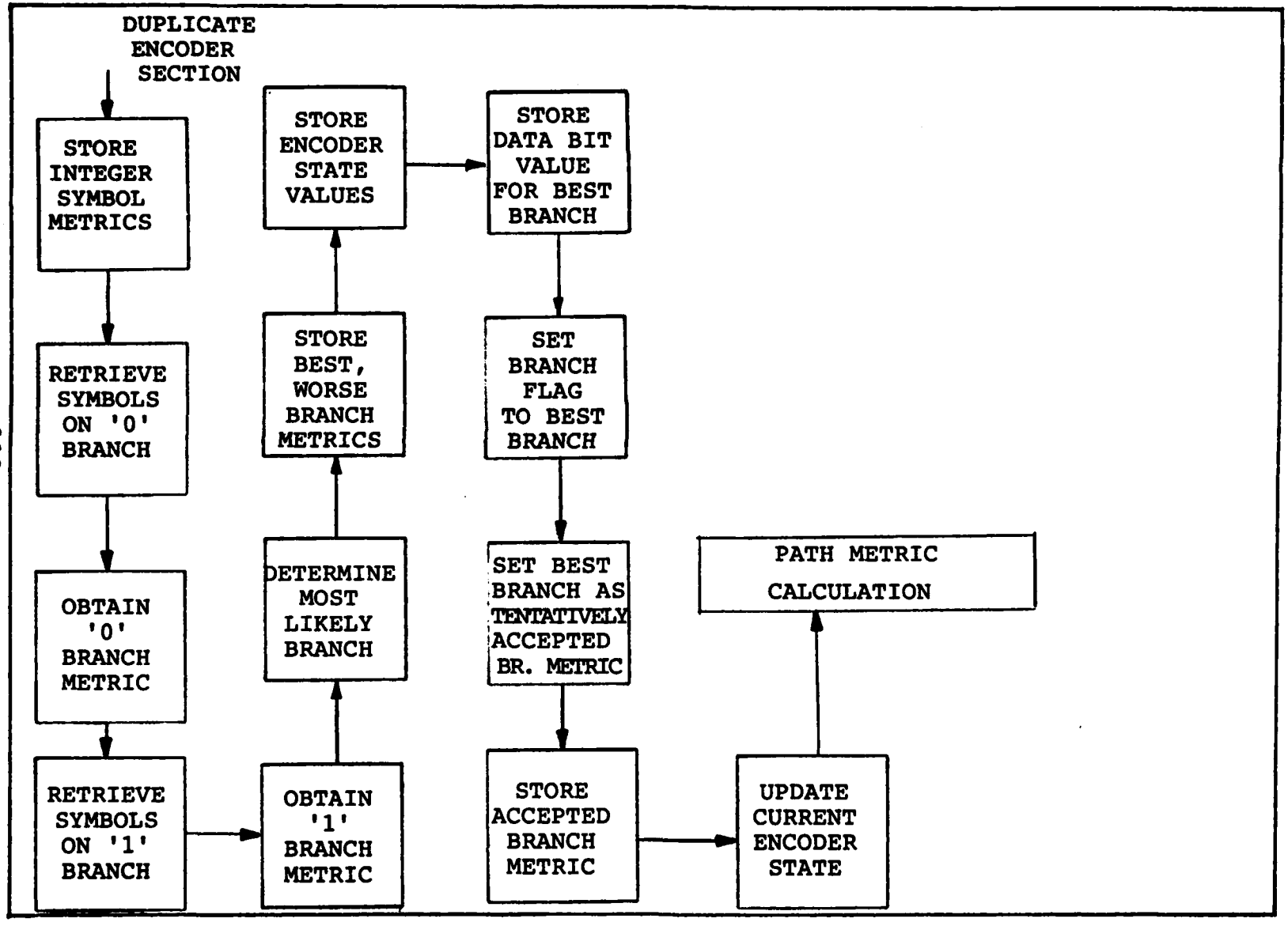


Figure 32. Flow Diagram for the Calculation of Branch Metrics.

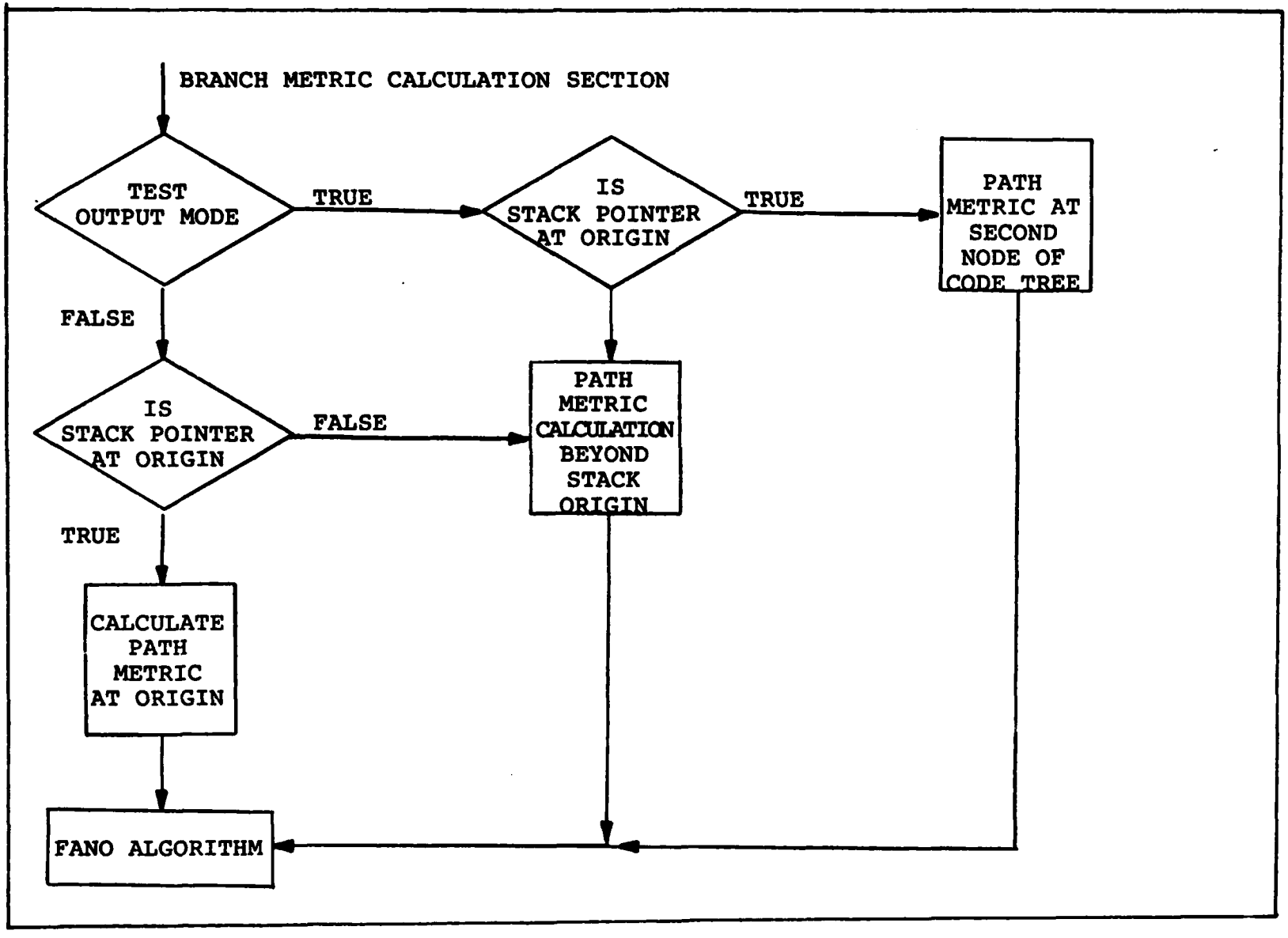


Figure 33. Flow Diagram of the Path Metric Calculation Section.

$$N = B_{comp}/B_R \tag{40}$$

where, B_{comp} = number of bit computations and B_R = total number of decoded bits received in a link simulation.

The amount of time spent in a search is nearly proprotional to the value of N, i.e.,

$$T_C \sim NT_H \tag{41}$$

where T_C is the average time spent in a search and T_H is a hardware or software sensitive value related to the cycle time for one complete search in the sequential decoder (Ref 2:13). Eq (41) above, however, is only approximately true since the $(n - 1) \rightarrow n$ operation in the search loop can be performed more than once in succession. Blustein and Jordan found in practice,

$$T_C \sim 1.1\ NT_H \tag{42}$$

closely approximated the average time spent in a search (Ref 2:13).

Sequential Decoder Verification Analysis

The sequential decoder subroutine was tested in a 2400 bps, DBPSK link simulation for the AWGN channel. A rate 1/2, constraint length 30 convolutional code described in Chapter III was incorporated in the simulation. The sequential decoder used 3-bit soft decisions and a decoder path memory length, DPML, of 133 bits. That is, an output bit

decision was made after 4.43 constraint lengths of decoded data had been processed.

Verification of the 2400 bps link sequential decoder operation is based upon a comparison between its performance and two other sequential decoders presented by Forney (Ref 10:57). Figure 34 illustrates the performance of the three sequential decoders in terms of undetected decoded bit error rate (BER) versus the received bit-to-noise density ratio (E_b/N_o). The two Forney sequential decoder performance results are for a 5 Mbps and a 50 Kbps link. Both of these links utilize rate 1/2 convolutional codes, coherent binary phase-shift keying (CPSK) and hard decisions. However, the constraint length of the encoders and DPML of the sequential decoders are unknown.

The important similarity in comparing these sequential decoders is the slope of the BER vs E_b/N_o performance curves. Note the 2400 bps link sequential decoder performance is similar to Forney's decoders in that it has a steep sloping curve that approaches some theoretical limit. As shown by Figure 34, Forney's sequential decoders approach a theoretical limit in performance of approximately 4 to 5 dB: while the sequential decoder simulated here approaches a theoretical limit of 7 dB.

Although the simulated sequential decoder utilizes soft decisions versus the hard decisions implemented by the Forney decoders, the 2 - 3 dB difference in theoretical limits of

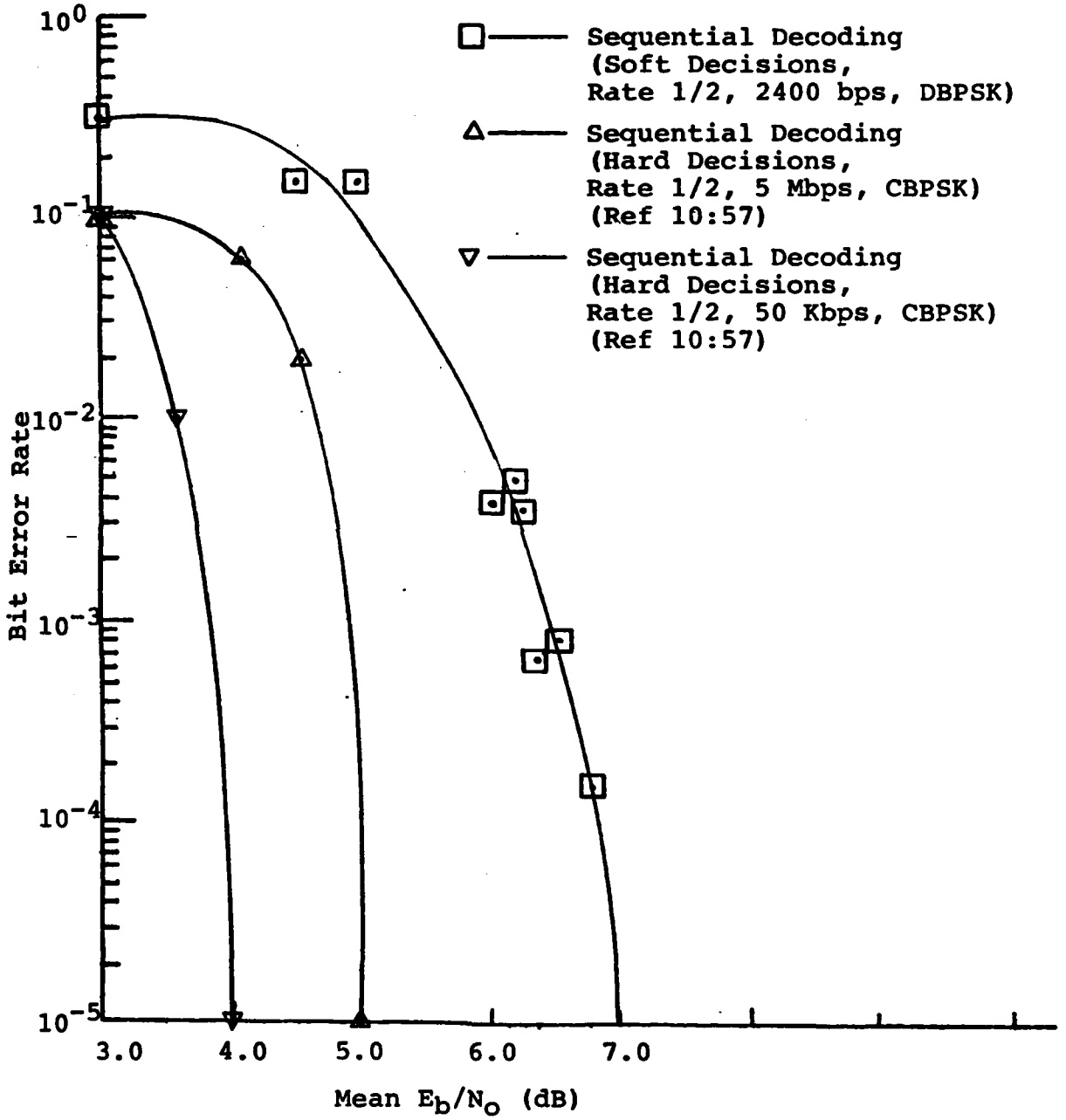


Figure 34. Sequential Decoder Performance in Additive White Gaussian Noise.

performance is largely due to the demodulation technique which is used. CPSK demodulation is superior to DBPSK in AWGN; that is, for a given E_b/N_o, the CPSK demodulator inputs fewer symbol errors to the sequential decoder than does the DBPSK demodulator. In the AWGN channel, CPSK demodulation can provide a couple of dB gain in performance over DBPSK demodulation.

To further verify the sequential decoder's operation, it is evident from Figure 35 that the undetected decoded bit errors of the simulated sequential decoder occur in widely separated noisy bursts. This characteristic was verified by Heller and Jacobs (Ref 16:847) in their performance assessment of sequential decoders for the AWGN channel. A brief discussion of the source of the undetected bit errors will be presented in a following chapter. Additional sequential decoder verification is substantiated by the sequential decoder performance results provided in Chapter VIII.

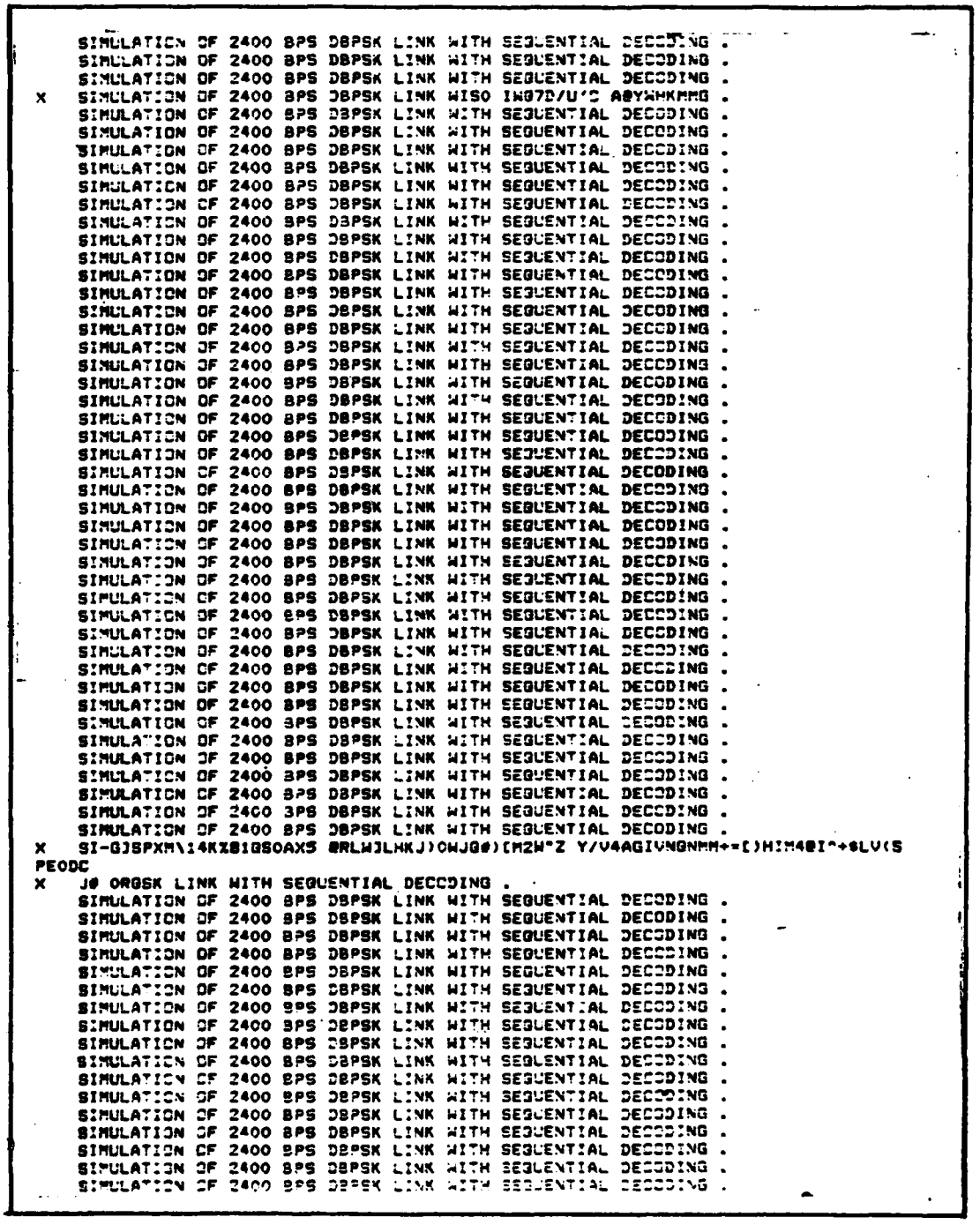


Figure 35. Received Message Error Characteristics of a Sequentially Decoded Link in AWGN.

VI. Viterbi Decoding

A detailed discussion of the Viterbi decoding algorithm will not be presented. The interested reader is referred to papers by Viterbi (Ref 25:751-772), Heller and Jacobs (Ref 16: 835-848) and Forney (Ref 11:268-278). A description of the Viterbi decoder used in the 2400 bps, DBPSK link simulations of this thesis was written by Allen Michelet of Mission Research Corporation for the Defense Nuclear Agency (Ref 22: 1-93).

To provide quantitative results for comparison of Viterbi decoding for short constraint length convolutional codes and sequential decoding for longer constraint length codes, a number of link simulations were performed using the Viterbi decoder. These simulations were done for both the additive white Gaussian noise (AWGN) and disturbed channels. To understand the significance of the comparitive analysis between these two decoding schemes, the characteristics of the Viterbi decoding algorithm are defined below.

Viterbi decoding is a maximum likelihood decoding technique for convolutional error correction codes. The Viterbi algorithm tries to reconstruct the trellis path, refer to Figure 12 in the convolutional encoding chapter, in some optimum fashion. In essence, the algorithm compares the received symbol sequence (which may have errors) to all possible paths in the trellis diagram. That path which is

"closest" to the received sequence is chosen as the sequence most likely transmitted.

Like the sequential decoder, the Viterbi decoder is a memoryless (random error) channel correcting device. If enough received symbol errors occur in a short space of time (burst) an incorrect path in the trellis diagram can be chosen. As a result, it can take some time before the decoder is aligned on the correct path again. In terms of performance the Viterbi maximum-likelihood algorithm is the optimum method of decoding convolutional codes for the memoryless channel, (Ref 25:751-772). However, since the computational complexity of Viterbi decoding increases expotentially with the constraint length, Viterbi decoding as a practical technique is limited to relatively short constraint length codes, $K \leq 10$.

The distance measure of metrics used between the received symbol sequence and various paths through the trellis diagram are the same metrics which can be used by the sequential decoder. Specifically, the Viterbi decoded link simulations utilized 3-bit (8-level) soft decisions bases upon integer symbol metrics. The Viterbi decoding algorithm uses these integer symbol metric values to find the path through the trellis diagram which has the smallest distance from the received sequence. The integer symbol metric table used for

Table VIII

Integer Code Symbol Metrics for a Rate 1/2 Convolutionally Encoded Link with 3-bit (8-Level) Receiver Quantization

		Quantization Level							
		0	1	2	3	4	5	6	7
Code Symbol	0	7	6	5	4	3	2	1	0
	1	0	1	2	3	4	5	6	7

the convolutionally encoded link simulations is shown in Table VIII.

The Viterbi decoding algorithm can make a bit decision after one constraint length of code symbols have been received; however, like the Fano sequential decoding algorithm, the decoder paths are stored back in time for 4 to 5 constraint lengths. In the Viterbi decoder simulations, the output bit decision is delayed by 31 bits. That is, the decoder path memory length (DPML) is 31. As each new symbol pair is received, the decoder bit 31 time intervals back on the most probable path is released.

VII. Interleaver/Deinterleaver Implementation

Convolutional codes in conjunction with soft-decision Viterbi or sequential decoding are used in digital satellite communication links to provide a random error correction capability. These forward error correction techniques can be quite powerful for the memoryless (uncorrelated error) channel. In a signal scintillation fading channel, however, demodulation errors tend to occur in burst during signal fades. The resulting nonrandom error patterns can significantly degrade decoding performance. Interleaving the encoded data and then deinterleaving the demodulated symbols provides a means of randomizing the errors so as to restore the error correction capability of Viterbi and sequential decoding. Previous assessments in the nuclear scintillated environment show DBPSK links with coding and interleaving have better performance than those without (Ref 12:13-34).

Convolutionally encoded DBPSK satellite communication links require somekind of interleaving even for the AWGN channel. DBPSK demodulated symbol errors typically occur in pairs which destroy the channel error randomness required by the Viterbi and sequential decoders. Thus, interleaving was incorporated in all of the encoded 2400 bps link simulations. Figure 36 is a simplified illustration of the interleaving and deinterleaving process.

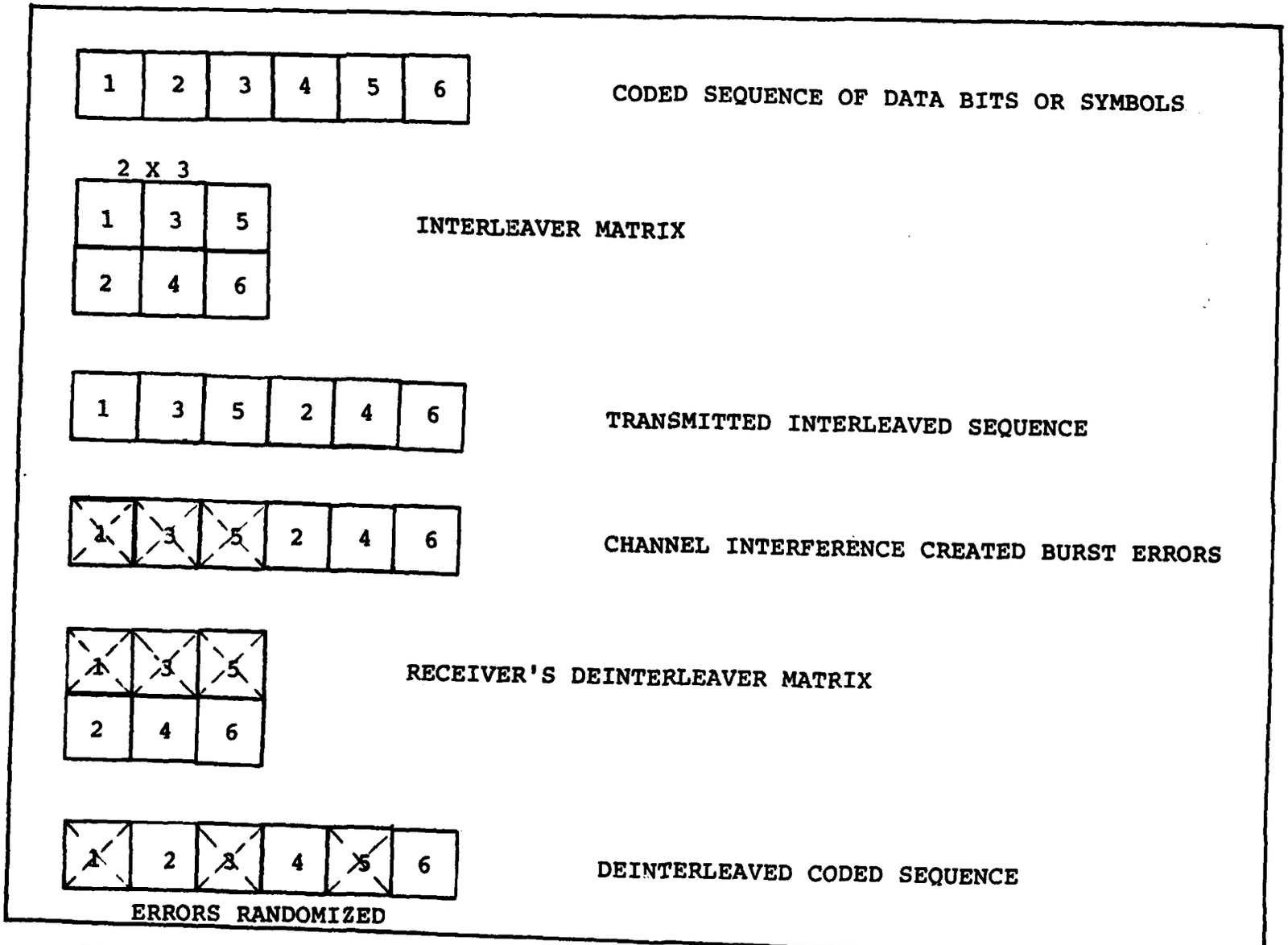


Figure 36. Simplified Example of Interleaving/Deinterleaving Process of Randomizing Errors.

Block interleaving and synchronous interleaving are two techniques for randomizing demodulated symbol errors. Both techniques are described in detail in a technical report by Bogusch and Guigliano (Ref 4:180-185). The former method which entails inserting the encoded data into an array in one order, and then reading the data out of the array in a different order, was not used. Synchronous interleaving, described in the next section, was implemented in the link simulations.

The principal advantage of synchronous interleaving is reduced ground and/or spacecraft memory storage requirement. Compared to a block interleaver with the same interleaving depth (separation distance between adjacent encoded symbols) a synchronous interleaver reduces the amount of receiver memory by more than a factor of two. Hence, with a given amount of available receiver memory a synchronous interleaver allows more than twice as much interleaving depth for increased protection against signal fading conditions. In addition, synchronous interleaving is particularly well suited to convolutionally encoded links (Ref 4:164).

Unfortunately, interleaving is not without its drawbacks. The major drawback is the time delay in message processing at the receiver. Time delays generated from interleaving are critical factors when designing a digital satellite communications link for processing 2400 bps (medium data rate) information.

Synchronous Interleaving and Deinterleaving

A synchronous interleaver is implemented by arranging its memory storage into a number of unequal length shift register stages, together with controlling logic to perform the gating and shifting operations. Synchronous interleavers operate continuously (after an initial delay) in such a way that one interleaved symbol is read out each time an input is shifted into the interleaver.

The synchronous interleaver configuration implemented in all of the link simulations is described in a paper by Ramsey (Ref 24:334-335) as a Type IV (n_2, n_1) interleaver. The definition of the two parameters n_1 and n_2 for a synchronous (n_2, n_1) interleaver can be stated as follows:

> No contiguous sequence of n_2 symbols in the reordered (interleaved) sequence contains any symbols that were separated by fewer than n_1 symbols in the original ordering (Ref 4:165).

The parameter n_1, determines the minimum separation that two adjacent demodulation errors will have at the decoder input after deinterleaving. For example, with Viterbi or sequential decoding, it is desirable to separate correlated errors by at least 4 to 5 code constraint lengths. Therefore, the value of n_1 is related to decoder memory. With a rate R, and constraint length K convolutional code, the value of n_1 should satisfy the following requirement:

$$n_1 > 4\frac{K}{R} \tag{43}$$

where the factor of 4 is the minimum number of code constraint lengths over which errors should be uncorrelated for optimum decoder operation.

Parameter n_2 determines the minimum separation of adjacent encoded input symbols in the output transmission sequence. Thus for two adjacent symbol errors to be present at the decoder input, an error burst due to signal fading would have to encompass n_2 symbol bit periods. Therefore, the value of n_2, commonly referred to as the interleaver depth, is related to the maximum expected fade duration. The fade duration is measured by the signal decorrelation time, τ_o. To provide sufficient interleaving depth so that fade durations corresponding to a specified value of τ_o are not likely to produce correlated errors at the decoder input, the value of n_2 should ideally satisfy the following requirement:

$$n_2 \geq (R_D/R)\tau_o \tag{44}$$

where R_D is the uncoded data bit rate and R is the code rate (Ref 4:162). Note, R_D/R is the modulation symbol rate.

This requirement on n_2 is, in practice, reduced somewhat without incurring extreme degradation of link performance. Bogusch and Guigliano (Ref 4:162), explain that the precise penalty for less than ideal interleaving depends on

the type and rate of the error correction code, the desired level of performance and available link margin. An approximation of the maximum value of τ_o for which a synchronous interleaver can provide sufficient interleaving for a particular value of n_2 is given by:

$$\tau_o = 10n_2/(R_D/R) \text{ sec} \qquad (45)$$

That is to say, if $\tau_o \leq 10n_2/(R_D/R)$ sec degradation of link performance will be less than 3 db. The Type IV (n_2, n_1) synchronous interleaver configuration is described by Ramsey to be optimum when $n_2 > 2n_1$, and n_1 and n_2 are relatively prime (n_1 and n_2 must not contain any common factors).

The implementation of a Type IV (n_2, n_1) synchronous interleaver is illustrated in Figure 37. The algorithm for this type of interleaver can be explained as follows:

1. Shift all stages right one bit, insert new symbol into tap $n_1 - 1$ (farthest from output);
2. Shift only those stages following tap $n_1 - 2$ right one bit, insert new symbol into tap $n_1 - 2$;
3. For each k, $k = 1, 2, \ldots, n_1$, repeat in the same manner. Shift only those stages following tap $n_1 - k$ right one bit, insert new symbol into tap $n_1 - k$.

The value of n_2 is used with n_1 to determine the storage capacity of each shift register. The required

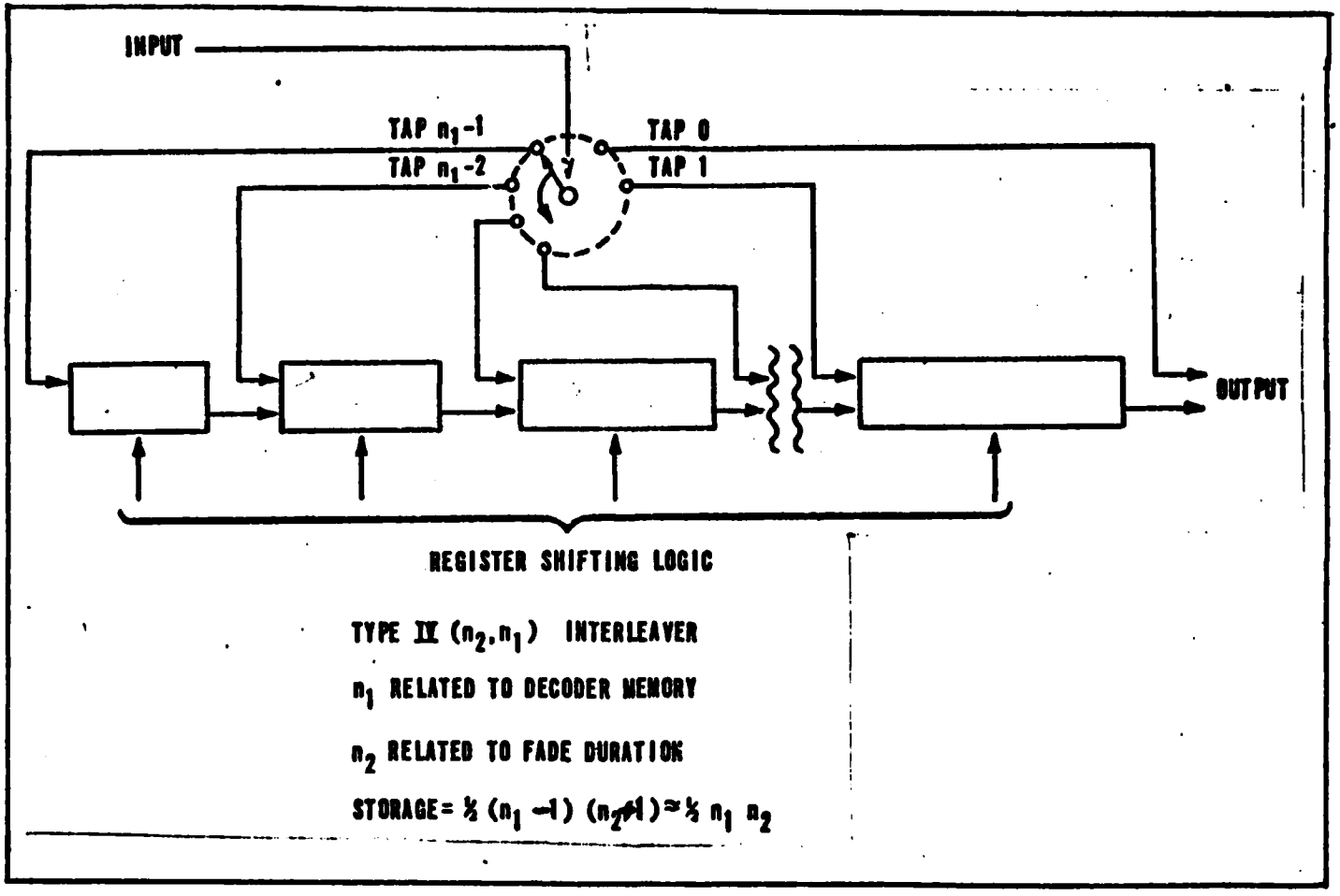


Figure 37. Example of a Synchronous Interleaver (Ref 4:166).

storage capacity for the left-most shift register is given by (Ref 4:167):

$$S_{n_1-1} = 1 + \left[\frac{n_2}{n_1} \right] \tag{46}$$

where [ ] denotes the truncated integer part of the enclosed expression. Eq (46) has been derived from the fact the left-most set of shift register stages following tap $n_1 - 1$ needs only enough storage capacity to contain one input symbol for each set of n_1 input symbols, or fraction thereof.

The shift register stages following tap $n_1 - 2$ require enough storage capacity for one input symbol for each set of n_1 symbols plus one symbol received from the preceding stage set of n_1 input symbols. Therefore, the required storage in this shift register is given by

$$S_{n_1-2} = 1 + \left[\frac{n_2}{n_1} \right] \tag{47}$$

The general expression for the storage capacity required by the set of shift register stages following tap $n_1 - k$ is given by

$$S_{n_1-k} = 1 + \left[\frac{kn_2}{n_1} \right] \tag{48}$$

where

$$1 \leq k \leq n_1 - 1$$

As shown in Figure 37, no shift register is needed following tap 0 (output of interleaver).

The function of the deinterleaver at the DBPSK receiver site is to invert the interleaving process so as to restore the demodulated symbol bits in their original sequence at the input to the decoder. The implementation of the deinterleaver for a Type IV (n_2, n_1) synchronous interleaver is shown in Figure 38, and the algorithm follows:

1. With the output connected to tap $n_1 - 1$, shift a new input symbol directly to output.
2. Extract output symbol from tap $n_1 - 2$, shift only those stages from input through this tap while inserting a new input symbol.
3. Decrement tap number by 1 and repeat. That is for $k = n_1 - 1, n_1 - 2, \ldots 1, 0$ extract output symbol from tap k, shift only those stages preceding this tap while inserting a new input symbol.

The number of stages required in each shift register of the deinterleaver is the same as that of the corresponding interleaver shift register taken in reverse order. A general expression of stages in the deinterleaver shift register preceding tap number k is given by

$$S_k = 1 + \left[\frac{(k+1)n_2}{n_1} \right] \quad , \quad 0 \leq k \leq n_1 - 2 \qquad (49)$$

An important consideration for determining the specific synchronous interleaver implementation to be used in a

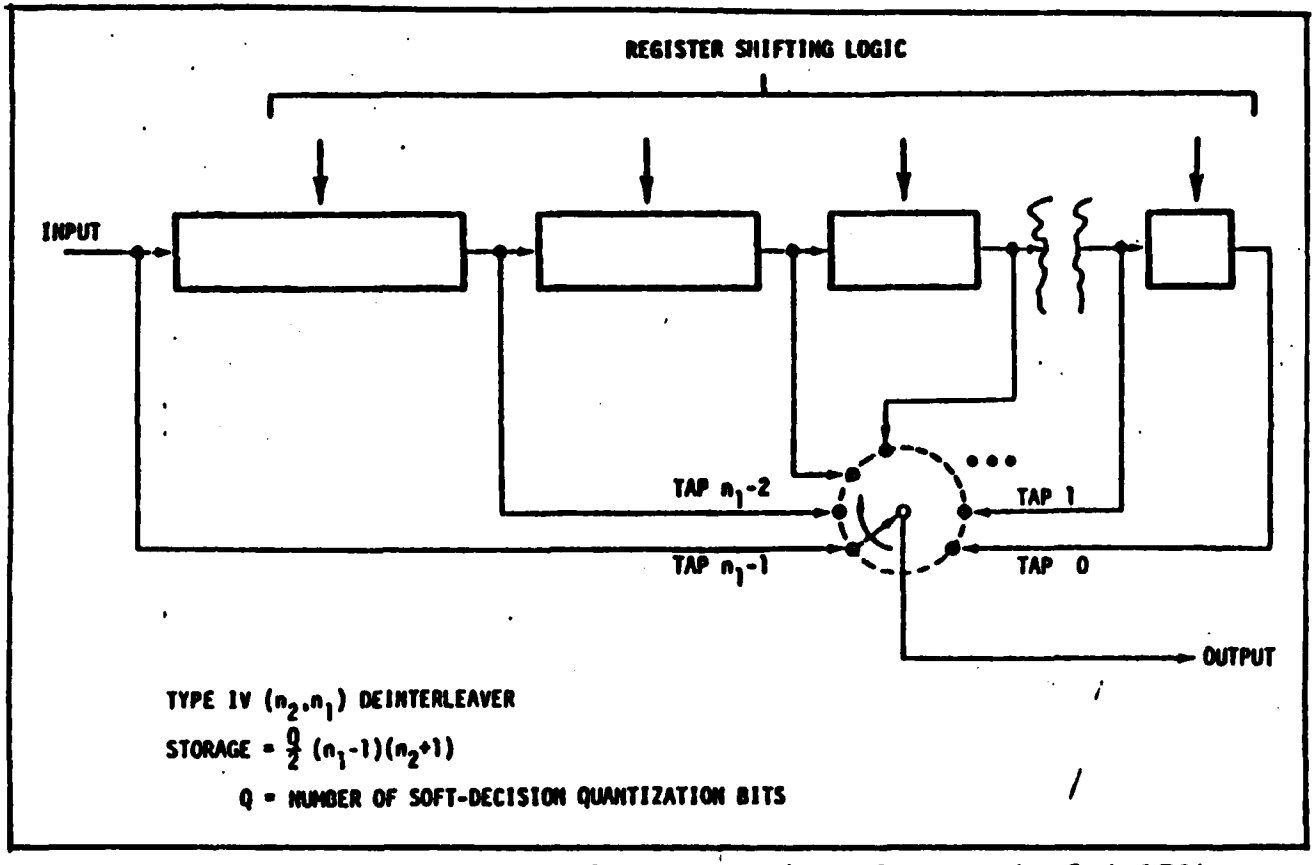


Figure 38. Example of Synchronous Deinterleaver (Ref 4:173).

satellite link is the total storage required by the interleaver and deinterleaver. The storage requirement for a Type IV (n_2, n_1) interleaver is given by (Ref 12:39):

$$S_I = 1/2\ (n_1 - 1)\ (n_2 + 1) \text{ bits} \tag{50}$$

The storage requirement for the deinterleaver used in the link simulations is equal to the storage requirement for the Type IV (n_2, n_1) interleaver, i.e., $S_D = S_I$. As indicated by Eq (50), the parameter values of n_2 and n_1 dictate the overall memory storage requirement for this type of synchronous interleaver at the transmitter and receiver.

Another important synchronous interleaver link design consideration is the total time delay associated with processing a single encoded bit of information through a Type IV synchronous interleaver and deinterleaver. This time delay is generated by the total storage requirement for the interleaver and deinterleaver (Ref 12:39) given by the equation:

$$D = S_I + S_D \text{ bits,} \tag{51}$$

where the total time delay in seconds can be expressed as:

$$D_S = \frac{D}{(R_D/R)} \tag{52}$$

$$= \frac{D}{4800}$$

for a 2400 bps, rate 1/2 encoded link.

Thus, it follows the parameters n_2 and n_1 not only specify the Type IV interleaver implementation and memory storage requirements but the link's interleaving time delay as well.

Synchronous Interleaver Implementation for the Viterbi and Sequential Decoded Link Simulations

A Type IV (133, 61) synchronous interleaver was implemented for both the Viterbi and sequential decoded link simulations. The interleaver parameter values, $n_2 = 133$ and $n_1 = 61$, provide optimum interleaving for the Viterbi decoded link and a certain degree of sub-optimum interleaving for the sequential decoded link. The interleaver and deinterleaver storage requirements and resultant time delays are listed in Table IX. These storage requirements and time

TABLE IX

Synchronous Interleaver Parameter for the Viterbi Decoded Satellite Communication Link

Code Constraint Length K	7
Type IV (n_2, n_1) Interleaver	(133,61)
Interleaving Period (sec)	0.027708
Total Delay (bits) (D) = $(n_1-1)(n_2+1)$	8,040
Total Delay (sec) (D_S)(Rate 1/2 Code)	1.675
Total Transmitter Storage (bits)	4,020
Total Receiver Storage (bits) (3-bit Soft decision decoding)*	12,060

*The link simulations incorporated 3-bit quantization of the demodulated output.

delays were derived using Eqs (46) through (52). Using the approximation given by Eq (45), it was found the above interleaver could provide sufficient interleaving for a slow fading channel with a maximum decorrelation time,

$\tau_o \cong 0.28$ sec .

The link simulations incorporating sequential decoding may have been subject to a small degree of sub-optimum interleaving due to the value of n_1 . A 'desirable' value of n_1 , specified by Eq (43), was not used in these simulations because of the large impractical interleaving/deinterleaving time delays and transmitter/receiver storage requirements for an actual 2400 bps link with rate 1/2, large constraint length convolutional encoding. A comparison of the delay times and storage requirements associated with constraint lengths 25, 30, and 7 encoded systems for optimum interleaving are listed in Table X. The values of n_1 and n_2 for the optimum synchronous interleaving cases were derived using Eqs (43) and (44).

TABLE X

Comparison of Synchronous Interleaver Parameter Values for Various Code Constraint Lengths

Code Constraint Length K	30	25	7
Type IV (n_2, n_1) Interleaver	(541, 269)	(499, 223)	(133, 61)
Interleaved Period (sec)	0.11	0.10	0.027708
Total Delay (bits) $(D) = (n_1-1)(n_2+1)$	145,256	111,000	8,040
Total Delay (sec) (D_S) (Rate 1/2 Code)	30.26	23.125	1.675
Total Transmitter Storage (bits)	72,628	55,500	4,020
Total Receiver Storage (bits) (3-bit soft decision decoding)*	217,884	166,500	12,060

*The link simulations incorporated 3-bit quantization of the demodulated output.

VIII. Sequential vs Viterbi Decoder Performance for the AWGN and Scintillated Channel

This chapter examines and compares the performance of sequentially decoded long constraint length codes versus a Viterbi decoded short constraint length code for the 2400 bps, DBPSK, satellite communications link. The performance results of the Viterbi and sequentially decoded links are presented for the AWGN and scintillated channels. Results are presented in terms of bit error rates which have been measured from detailed digital link simulations using the AFWL FSK-PSK Code discussed in Chapter III. Sequentially decoded link performance results should be considered as optimal because link simulations do not demonstrate the effect of overflow. However, a measure of the average sequential decoder computational requirements per decoded bit are presented.

The maximum likelihood Viterbi decoding of a rate 1/2, constraint length 7 convolutional code provides the baseline of comparison for sequential decoding of 1/2, constraint length 25 and 30 codes. A detailed discussion of the $R = 1/2$, $K = 7$; $R = 1/2$, $K = 25$; and $R = 1/2$, $K = 30$ convolutional codes is presented in Chapter IV.

Some of the receiver design parameters which are held constant for the Viterbi and sequential decoded link simulations are given in Table XI. The decoder path memory lengths,

TABLE XI

Constant Receiver Design Parameters

Type of PSK modulation	DBPSK
PSK data bit rate	2400 bps
PSK code symbol rate	4800 bps
A/D sampling rate	9600 Hz
Carrier frequency	20 GHz
No. soft-decision bits	3
Convolutional encoding rate	1/2
Synchronous interleaver	Type 4 (n_2=133, n_1=61)

TABLE XII

Code Constraint Length and Corresponding Decoder Path Memory Lengths for the Various Viterbi and Sequential Decoded Link Implementations

Decoder	Code Constraint Length (K)	Decoder Path Memory Length (DPML)	DPML K
Viterbi	7	31	4.43
Sequential #1	30	133	4.43
Sequential #2	25	111	4.44

DPML, for the sequential decoder implementations were set so that the ratio of DPML per code constraint length (K) would be the same as that given for the Viterbi decoded link. This DPML/K relationship is illustrated in Table XII. Therefore, all three decoders have approximately the same relative number of decoded bits that must be processed before a final bit decision can be outputted from the decoder.

In addition to the bit error rate performance results, implementation trade-offs between Viterbi decoded and sequentially decoded links are presented for both the undisturbed and signal scintillation conditions. Implementation considerations for the Viterbi and sequential decoders are discussed in terms of decoder bit processing delays.

Performance of Sequential and Viterbi Decoding in AWGN

Figure 39 and Table XIII show the simulation results for the Viterbi decoder and the two sequential decoders in AWGN. Performance results are given in terms of bit error rate versus E_b/N_o. As illustrated in Figure 39 the Viterbi decoded $R = 1/2$, $K=7$ code outperforms the sequentially decoded $R = 1/2$, $K = 30$ and $R = 1/2$, $K = 25$ codes for bit error rates $\geq 10^{-5}$. Link assessments were not performed for bit error rates $< 10^{-5}$ due to the extensive amount of computer time to obtain meaningful statistical results. The bit error probability performance curves of all three convolutional coding systems are rather steep at 10^{-5}. Therefore, it isn't immediately apparent if either sequentially decoded system would outperform the Viterbi decoder as they approach their theoretical limits at lower values of bit error rate.

Considering the two sequentially decoded system bit error probability curves, the $K = 30$ code has a distinct advantage over the $K = 25$ code for E_b/N_o between 4.0 and

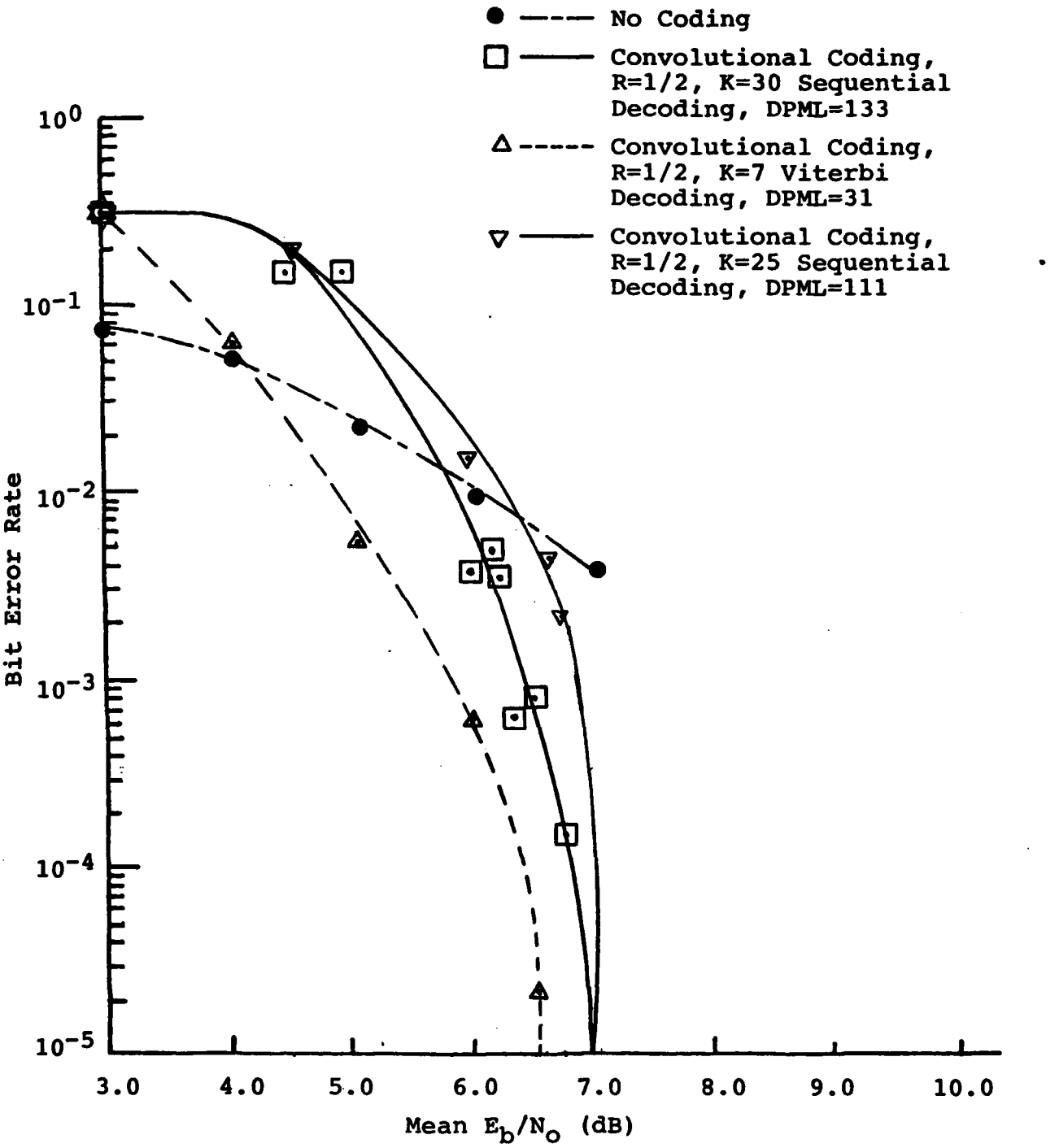


Figure 39. Sequential and Viterbi Decoded Link Performance in AWGN.

TABLE XIII

Simulation Data for 2400 bps, DBPSK Link in AWGN

Mean E_b/N_o (dB)	Total	Bit Error Rate	Ave. No. of Computations per Bit
		Decoded Bits	
Convolutional Coding,R=1/2,K=7-Viterbi Decoding,DPML=31			
3.0	1,080	3.037×10^{-1}	N/A
4.0	1,080	6.111×10^{-2}	N/A
5.0	1,080	5.556×10^{-3}	N/A
6.0	100,158	6.290×10^{-4}	N/A
6.5	500,040	2.400×10^{-5}	N/A
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=133			
3.0	36	3.056×10^{-1}	2,699.300
4.5	25,200	1.516×10^{-1}	222.530
5.0	50,040	1.503×10^{-1}	129.855
6.0	100,080	3.777×10^{-3}	5.897
6.15	300,240	5.143×10^{-3}	4.867
6.25	100,080	3.417×10^{-4}	5.100
6.375	500,040	6.379×10^{-4}	0.950
6.5	500,040	8.419×10^{-4}	1.080
6.75	500,040	1.480×10^{-4}	0.546
7.0	500,040	0	0.230
Convolutional Coding,R=1/2,K=25-Sequential Decoding,DPML=111			
3.0	54	2.963×10^{-1}	278.800
4.5	20,160	2.008×10^{-1}	248.110
6.0	100,080	$1,469 \times 10^{-2}$	8.300
6.625	400,320	4.676×10^{-3}	1.280
6.75	500,040	2.126×10^{-3}	2.140
7.0	600,120	0	0.120

7.0 dB. This is primarily due to the difference in constraint lengths and decoder path memory lengths. Figure 39 illustrates, however, that the K = 30 code performance improvement over the K = 25 code decreases as E_b/N_o approaches 7 dB. In fact, both sequentially decoded links approach a theoretical limit in bit error performance at 7.0 dB. The significance of the simulation results for the sequentially decoded systems is not so much their statistical exactness but the overall trends of the bit error probability curves and the theoretical limits they approach. In addition, simulation results obtained at E_b/N_o values near the 7.0 dB theoretical limit were generated from computer link simulations that gave reasonable turn around time.

Comparing the Viterbi and sequentially decoded systems again, Figure 39 shows the Viterbi decoded K = 7 code provides bit error probability improvement over the uncoded link for E_b/N_o values greater than 4.0 dB. The sequentially decoded K = 30 and K = 25 codes, however, provide bit error probability performance improvement over the uncoded link for E_b/N_o values of 6.0 dB and 6.625 dB respectively. Viterbi and sequentially decoded system performance starts out inferior to the uncoded link at low received bit energy-to-noise density ratios. The cause of the comparitive degraded performance of the coded links is largely due to the bursty or correlated nature of the demodulated symbol errors inputted to the decoders. Although the links are interleaved,

TABLE XIV

Viterbi Decoded Performance Gains of the R = 1/2, K = 7 Encoded Link Over the Sequential Decoded Links in AWGN

BER	R = 1/2, K = 30 Code	R = 1/2, K = 25 Code
10^{-3}	0.6 dB	1.0 dB
10^{-5}	0.5 dB	0.5 dB

the frequency of symbol errors at low E_b/N_o's appear clumped and thus destroy the random error-correction capabilities of all three of the decoders. The demodulated symbol errors then become more random or memoryless as the E_b/N_o increases.

Table XIV lists the Viterbi decoded link performance gain over the two sequentially decoded links for a bit error probability of 10^{-3} and 10^{-5}. For these bit error rates, performance gain of the coded systems differ, at most, by 1.0 dB.

Figure 40 illustrates the average number of computations versus E_b/N_o to decode a single bit of information. The significance of this computational value is that it accounts for the time delay induced by sequential decoders to produce a final bit decision.

Viterbi decoders essentially have a decoding delay in information bits equal to the decoder path memory length, DPML. The bit delay can be transformed into a time delay relationship for the Viterbi decoder implementation in this investigation by way of the following:

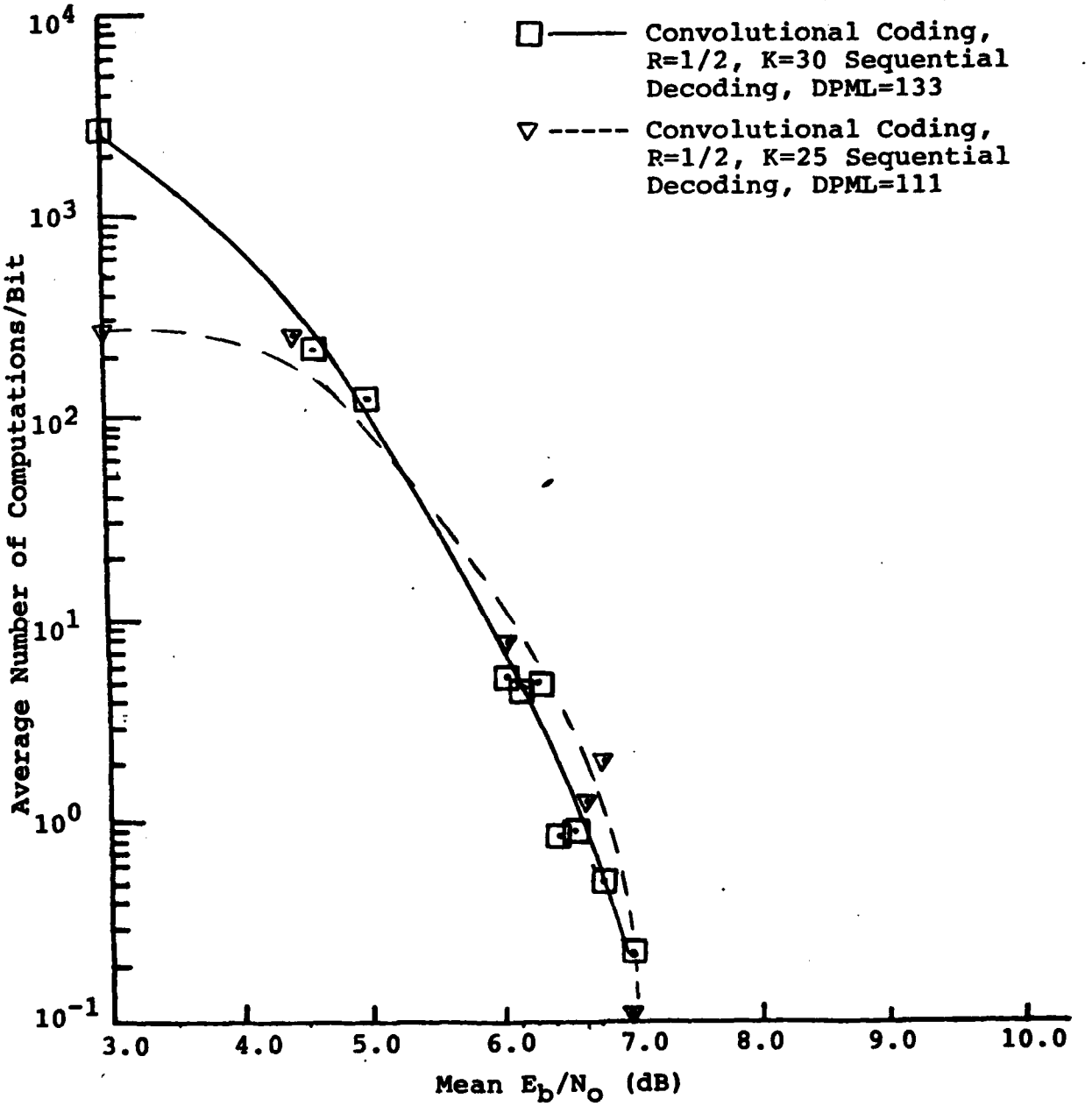


Figure 40. Average Number of Sequential Decoder Computations Required to Decode a Single Data Bit of Information in AWGN.

$$T_D = \frac{DPML}{R_D}$$

$$= \frac{31}{2400}$$

$$= 0.013$$

where

R_D = information data rate (bps)

T_D = decoder path memory length processing delay (sec)

while for the sequential decoder, the average time delay for processing a single bit of data is given by

$$T = T_D + T_C \tag{53}$$

where T_C equals the decoder computational delay approximated by Eq (41) in Chapter V. The average time delay associated with the $K = 30$ sequentially decoded system is given by

$$T = \frac{133}{2400} + T_C$$
$$= 0.055 + T_C \tag{54}$$

and the average time delay for the $K = 25$ sequentially decoded system is found to be

$$T = \frac{111}{2400} + T_C$$
$$= 0.046 + T_C \tag{55}$$

Hence, decoder bi processing delays are a viable consideration in determining the implementation trade-offs between the Viterbi and sequentially decoded systems. In addition, the average time delay given by Eq (53) is useful as a starting point for the communication systems analyst concerned about potential overflow problems for an actual hardware/software sequential decoder implementation.

Returning to Figure 40, the average number of computations per bit decreases as E_b/N_o increases. Note the sequentially decoded system with code constraint length 25 and decoder path memory length 111 made bit decisions based upon a fewer number of searches compared to the other sequential decoded link for $E_b/N_o < 6$ dB . This is because the former sequential decoder can only search back 111 code tree nodes in attempt to follow the correct search tree path, while the other decoder can search back 133 nodes. Above 6 dB, both decoders search back into the code about the same distance.

The received message characteristics of the Viterbi and sequentially decoded links are shown in Figures 41, 42, and 43. Viterbi errors occured in short burst, engulfing 1-5 message characters, and were randomly spread in any one link simulation. In addition, the messages or lines received in error were still easily interpreted.

Figure 42 shows the sequentially decoded errors occured in more widely separated burst entailing a larger number of

```
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DEC/1ING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION OF 2460 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION OF 2400 BPS DBPSK LINK WI>L VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION OF 2400 "7S DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION OF 2400 BPS LLEWK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X  SIMULATION /\ 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
```

Figure 41. Viterbi Decoded Message Errors in AWGN.

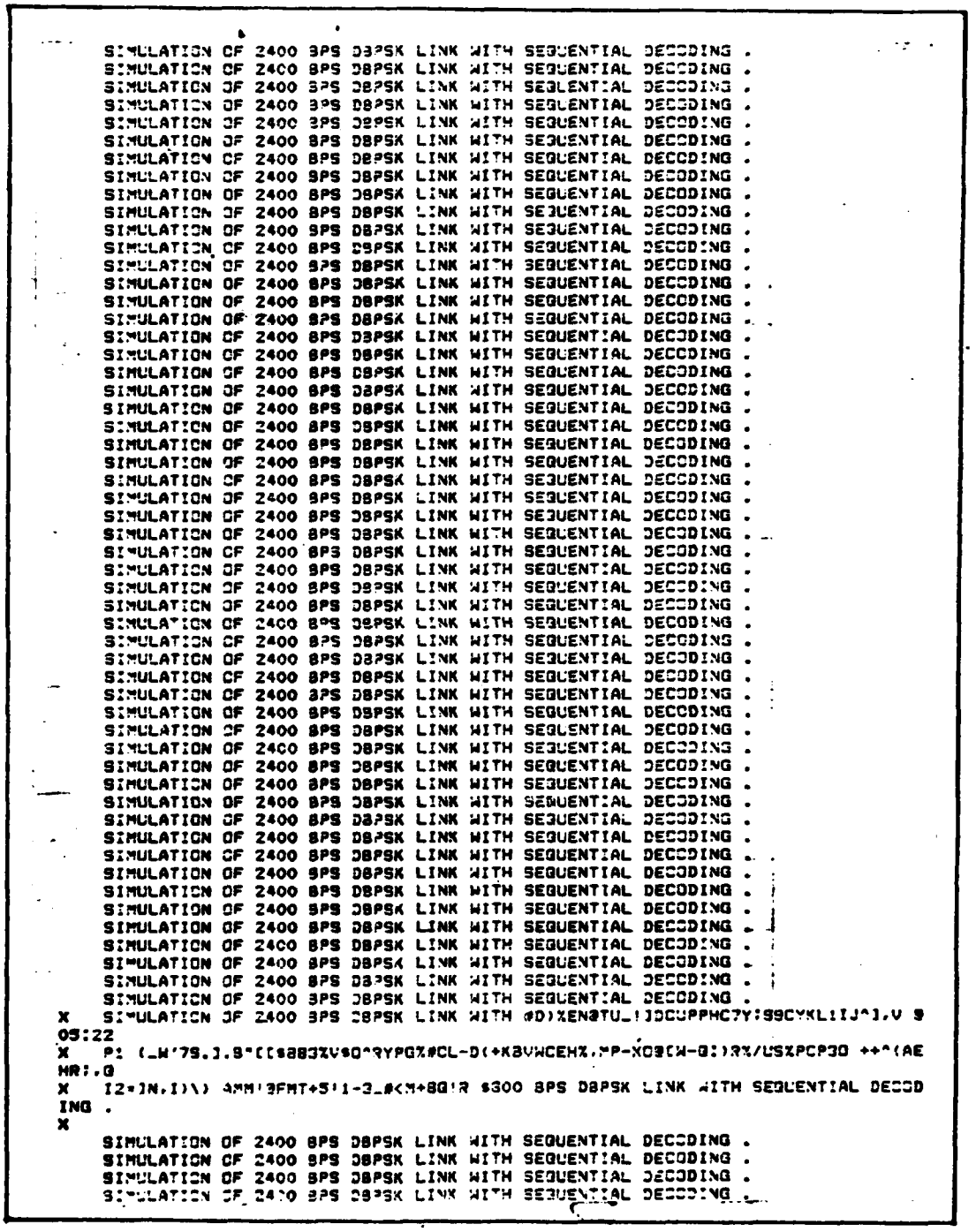


Figure 42. Sequential Decoded Message Errors in AWGN with K = 30 Code.

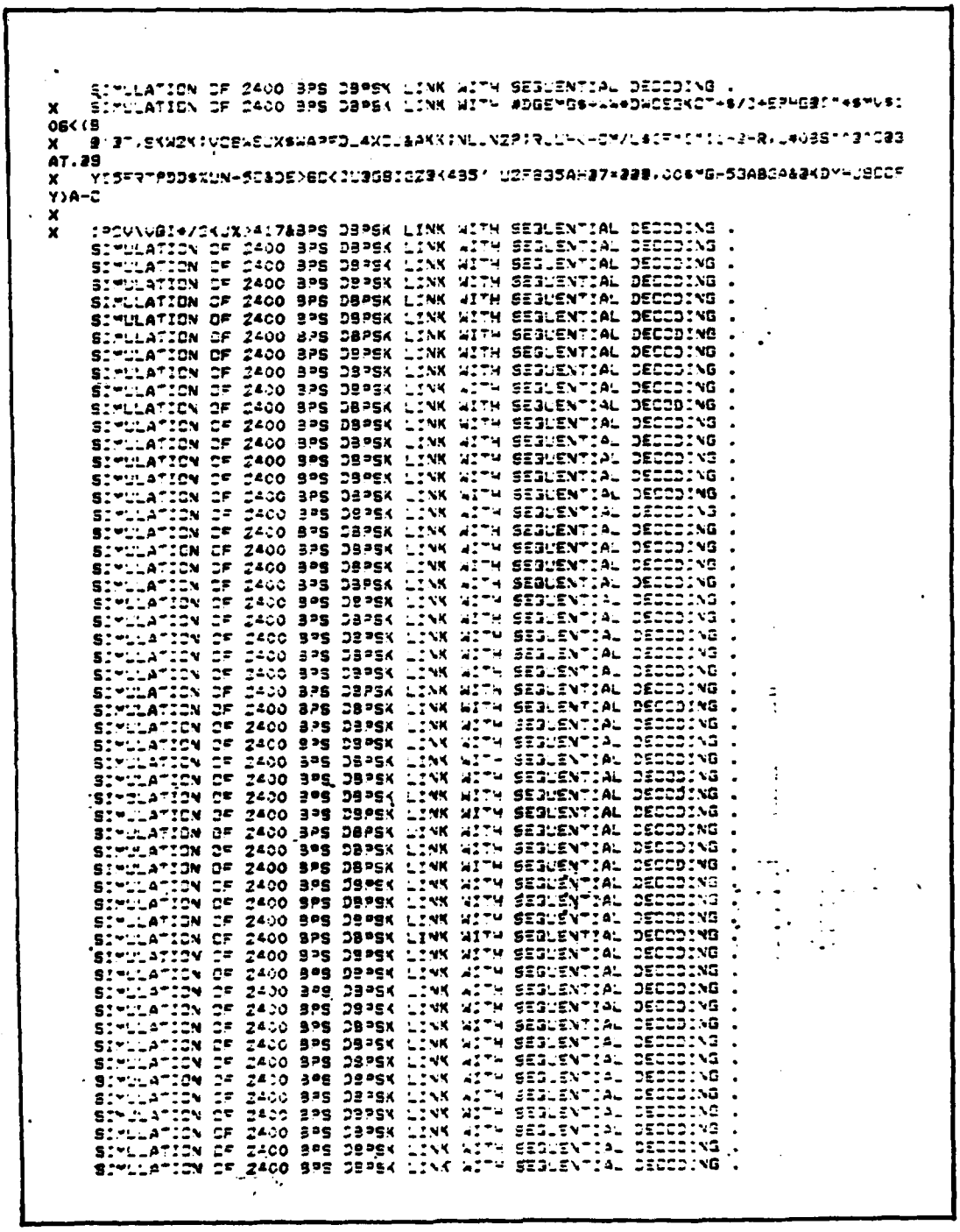


Figure 43. Sequential Decoded Message Errors in AWGN with K = 25 Code.

message characters. During these bursty periods the messages were totally uninterpretable until the decoder began straightening itself out. An interesting observation made during many of the link simulations was, for $E_b/N_o \geq 6$ dB , the sequentially decoded message came thru without errors over longer time periods than did the Viterbi decoded messages. Hence, sequential decoding looks promising for the satellite communications link requiring low message error rates. Message error probability of performance for sequential decoding versus Viterbi decoding, however, requires further investigation.

Undetected bit errors with sequential decoding are generally one of two types. The first source of error arises, when because of the finite size of the sequential decoder path memory length, the decoder is not permitted to search sufficiently far back in the code tree and change its mind about an earlier tentative decision. The sequential decoders implemented in this performance assessment can only go back as far as the origin of the search tree stack. A follow-on extension of the search tree stack may not correspond to the correct path at first but the decoder eventually finds the correct path as illustrated in Figure 43. These kinds of errors, which also create overflow problems for actual hardware/software sequential decoders, can be reduced in probability of occurrence by increasing the decoder path memory length (Ref 19:497).

The second source of errors occurs less frequently. An especially noisy received sequence can very closely approximate the correct transmitted sequence generated by the decoder's duplicate convolutional encoder. In these instances it is possible for several incorrect digits to be released without a large increase in the number of branches searched. As a result, the search tree path extends for the wrong path. Increasing the code constraint length can significantly decrease these kinds of undetectable errors. A more detailed discussion of sequential decoder error characteristics is explained by Wozencraft and Jacobs (Ref 29:446-454).

Figure 44 and Table XV show another aspect of the sequential decoder's bit error probability performance in AWGN. Illustrated here is the bit error rate performance for various decoder path memory lengths. The principal data points are those generated for the $K = 30$ convolutional code system operating with an $E_b/N_o = 6$ dB . Bit error probability improves dramatically as the decoder path memory length increases from 90 bits (3 times the constraint length) to 133 bits (4.43 times the constraint length). As the decoder path memory length approaches 150 bits the bit error rate performance begins to level off. The latter performance characteristic demonstrates why the decoder path memory length is typically set at 4 to 5 times the constraint length. In addition, two data points have been plotted for the sequentially decoded $K = 25$ code. These points indicate a similar

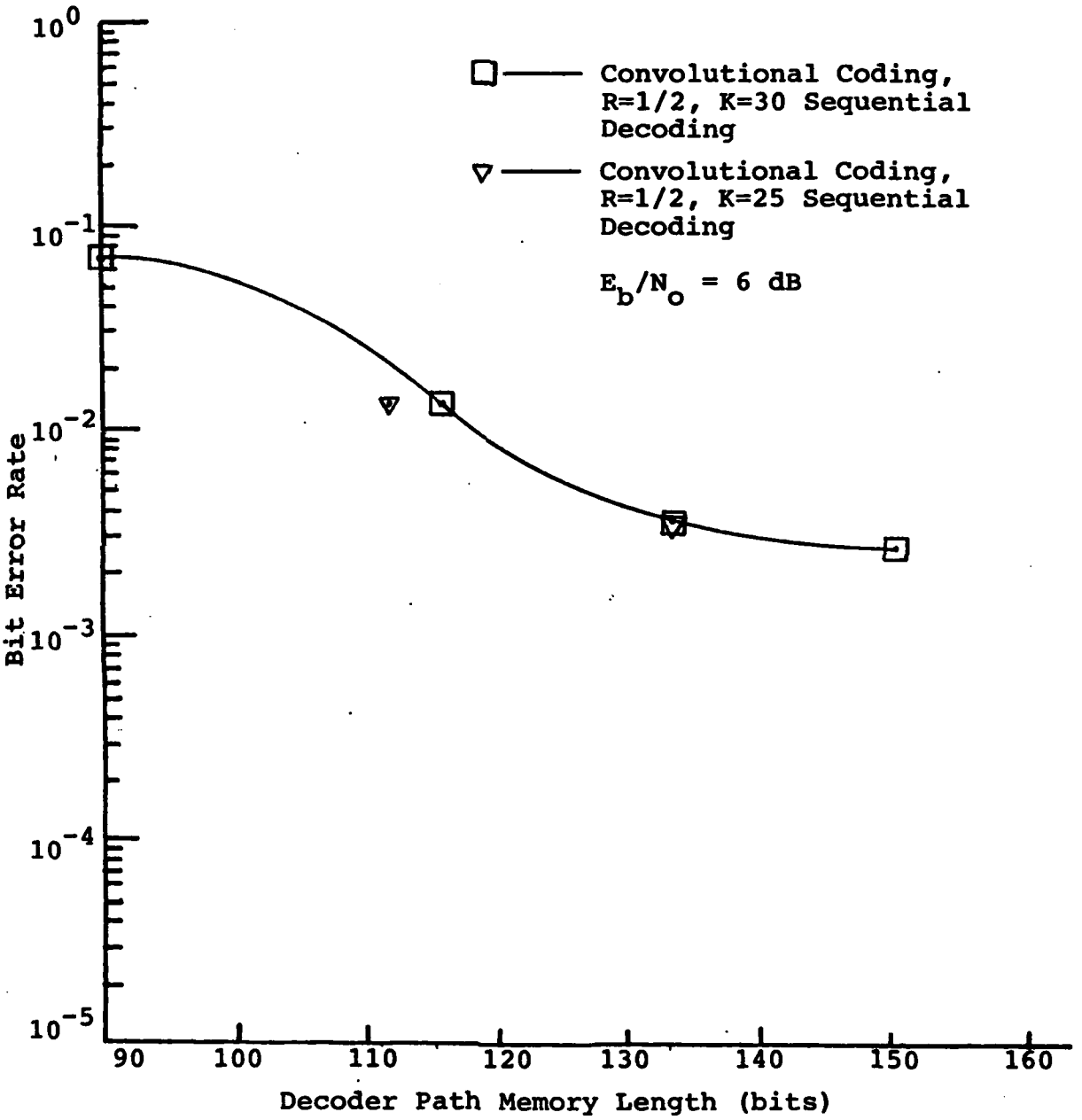


Figure 44. Performance of Sequentially Decoded Link with Various Decoder Path Memory Lengths.

TABLE XV

Simulation Data for 2400 bps, DBPSK Link in AWGN
For Various Decoder Path Memory Lengths

Constraint Length(K)	DPML	Multiple of K	Decoded Bits Total	Bit Error Rate
Rate 1/2 Convolutional Code - Sequential Decoding, E_b/N_o=6dB				
30	90	3.0	100,080	6.831×10^{-2}
30	115	3.83	100,098	1.490×10^{-2}
30	133	4.43	100,080	3.777×10^{-3}
30	150	5.0	100,080	2.858×10^{-3}
25	111	4.44	100,080	1.469×10^{-2}
25	133	5.32	100,080	3.777×10^{-3}

TABLE XVI

Simulation Data for 2400 bps, DBPSK Link in AWGN
for Two Constraint Lengths
E_b/N_o = 6 dB

Constraint Length(K)	Decoded Bits Total	Bit Error Rate
Rate 1/2 Convolutional Code - Sequential Decoding, DPML = 133		
18	100,080	1.070×10^{-2}
30	100,080	3.777×10^{-3}

bit error probability performance trend for the lower constraint length code.

Finally, Table XVI illustrates the change in bit error performance with varying constraint lengths. These results were derived using a constant decoder path memory length, DPML = 133 bits and an E_b/N_o = 6 dB . As the code constraint length increases from 18 to 30 a noticeable improvement in bit error probability performance occurred.

Performance of Sequential and Viterbi Decoding in Scintillation

The primary purpose of this section is to illustrate the performance characteristics of a sequential decoder operating in the scintillated channel. Secondly, the section compares that performance with the Viterbi decoded link and presents some potential implementation trade-offs between the two convolutional coded systems.

Performance results for the scintillated channel were obtained for three typical signal fading rates an EHF link would have to deal with after a high-altitude nuclear detonation. These fading rates correspond to a moderately fast, moderately slow and slow amplitude/phase scintillated fading channel. The time-selective or flat fading EHF channels were generated by a scintillated signal structure having a decorrelation time (τ_o) of 0.016, 0.05 and 0.16 seconds respectively. These values of τ_o are representative of fading period for a 20 GHz EHF link shown in Figure 4 of Chapter II.

An example of the scintillated signal structure which characterizes the channels simulated in this part of the analysis is depicted in Figure 2 of Chapter II. In addition, the scintillated link simulations were performed using the techniques discussed in Chapter III. Before moving on to a discussion of the convolutional coded systems response to signal scintillation, a general overview of the DBPSK demodulation response will be presented.

A significant amount of work has already been accomplished in determining the effects of signal scintillation on DBPSK demodulation of a 2400 bps link (Ref 7:127-149). The purpose is to summarize the DBPSK demodulation performance characteristics and describe the nature of the demodulated symbol errors that are sent on to the decoders implemented in the present analysis.

Amplitude fading and noise are the dominant sources of degradation at low fade rates (τ_o = 0.16 sec, for example) and low E_b/N_o. Demodulated symbol errors typically occur in clumps or burst. As the fade rate increases, however, DBPSK performance becomes progressively worse, from slow fading. At faster fade rates DBPSK performance degrades due to phase glitches generated by the phase scintillation effects more so than the amplitude scintillation in the channel.

DBPSK symbol decisions are based upon the average difference of the received baseband signal phase between sequential symbol periods. Symbol errors can occur when the

residual phase glitch changes the average signal phase difference between a sequential pair of symbols by more than 90 degrees (Ref 17:45). Therefore, a fast or even moderately fast fading channel (τ_o = 0.016 sec) simulated in this analysis produces DBPSK demodulation errors due to the loss of signal phase coherence over two consecutive symbol periods.

As previously mentioned in Chapter II, DBPSK demodulation performance in fast fading can be improved by increasing E_b/N_o, but this only works for so long in the 2400 bps case until a 'leveling-off" in bit error rate performance occurs. The leveling-off in performance is because the demodulated symbol errors which occur due to phase scintillation are not reduced by increasing signal power.

Unlike the bursty demodulated symbol error patterns in slow fading, the symbol errors in rapid fading are more random or uncorrelated. For the most part, in fast fading conditions, the channel appears memoryless. In moderately fast to moderately slow scintillation (τ_o = 0.05) the combined contribution of phase and amplitude scintillation effects increase. Referring now to the link simulations results for the Viterbi and sequentially decoded systems, Figure 45 and Table XVII show the bit error probability performance vs E_b/N_o for the moderately fast fading channel (τ_o = 0.016 sec). Again, the Viterbi decoded link outperforms the sequential decoded links for bit error rates $> 10^{-5}$.

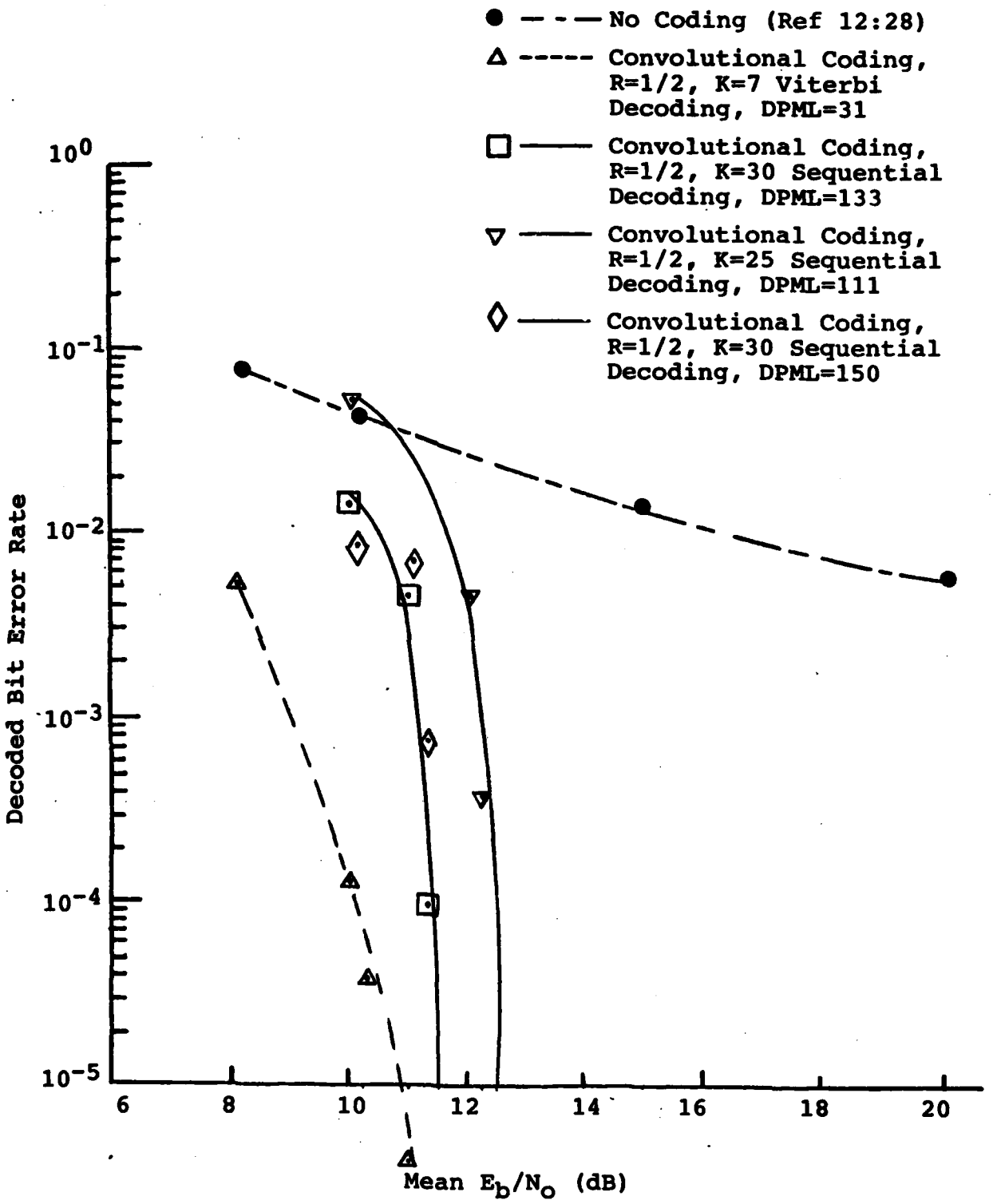


Figure 45. Encoded 2400 bps, DBPSK Link Performance in Moderately Fast Scintillation, τ_o = 0.016 sec .

TABLE XVII

Simulation Data for a 2400 bps, DBPSK Link in Moderately Fast Scintillation, τ_o = 0.016 sec

Mean E_b/N_o (dB)	Total Decoded Bits	Bit Error Rate	Ave. No. of Computations per Bit
Convolutional Coding,R=1/2,K=7-Viterbi Decoding,DPML=31			
8.0	50,040	5.256×10^{-3}	N/A
10.0	50,040	1.40×10^{-4}	N/A
10.25	400,320	4.247×10^{-5}	N/A
10.5	300,240	0	N/A
11.0	1,000,080	4.0×10^{-6}	N/A
12.0	100,182	0	N/A
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=133			
10.0	50,040	1.55×10^{-2}	7.79
11.0	110,160	4.630×10^{-3}	2.48
11.25	500,040	1.0×10^{-4}	0.17
11.5	500,040	0	0.123
12.0	110,160	0	0.105
Convolutional Coding,R=1/2,K=25-Sequential Decoding,DPML=111			
10.0	50,040	5.330×10^{-2}	16.63
12.0	110,160	4.520×10^{-3}	2.99
12.25	500,040	4.0×10^{-4}	0.23
12.5	500,040	0	0.07
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=150			
10.0	110,160	8.905×10^{-3}	4.85
11.0	110,160	7.244×10^{-3}	1.42
11.25	500,040	7.899×10^{-4}	0.99
11.5	500,040	0	0.12

A third sequential decoder with a larger decoder path memory length, DPML = 150 , was implemented to analyze changes in performance for the R = 1/2, K = 30 convolutionally encoded link. As illustrated in Figure 45, no significant changes occurred; in fact, this decoder essentially retained the same performance as the decoder with DPML = 133 . Clearly, the distance searched back into the code tree, in an attempt to follow the correct search tree path, was limited to 133 nodes.

The K = 30 code has a bit error probability performance gain over the K = 25 code of about 1.0 dB for this channel. Figure 45 shows the K = 30 sequentially decoded link approximates a theoretical limit of 11.5 dB, whereas, the K = 25 sequential decoded system approaches a theoretical limit of 12.5 dB. Thus both sequential decoders maintain their steep bit error probability performance curves from the AWGN channel case. However, significant increase in E_b/N_o is required to obtain comparable performance. The sources of undetected bit errors with sequential decoding that were previously discussed for the AWGN channel are the same for the scintillated channel. The difference is, for the scintillated channel, the undetected bit errors occur more frequently.

TABLE XVIII

Viterbi Decoded Performance Gain Over the Sequential Decoded Links in Moderately Fast Fading

BER	R = 1/2, K = 30 Code	R = 1/2, K = 25 Code
10^{-3}	.75 dB	1.75 dB
10^{-5}	2.25 dB	3.25 dB

Table XVIII lists the Viterbi decoded link performance gain over two of the sequentially decoded links. This table further demonstrates this maximum likelihood Viterbi decoded link using a shorter constraint length code still outperforms the longer constraint length sequentially decoded links implemented in this investigation. Lastly, the uncoded link performance results in Figure 45, as well as, Figures 46 and 47 were derived from previous analyses (Ref 12:25, 27, 31).

In the moderately slow signal scintillated fading channel (τ_0 = 0.05 sec) Figure 46 and Table XIX show the Viterbi decoded link has better bit error performance than either of the three sequential decoded systems. Again the difference in performance between the DPML = 133 and DPML = 150 sequential decoders is very small. In fact, all three sequential decoded bit error probability performance curves merge together to approach a theoretical limit of 13 dB. Differences in performance between the sequentially decoded K = 25 convolutionally encoded link and the K = 30 encoded link

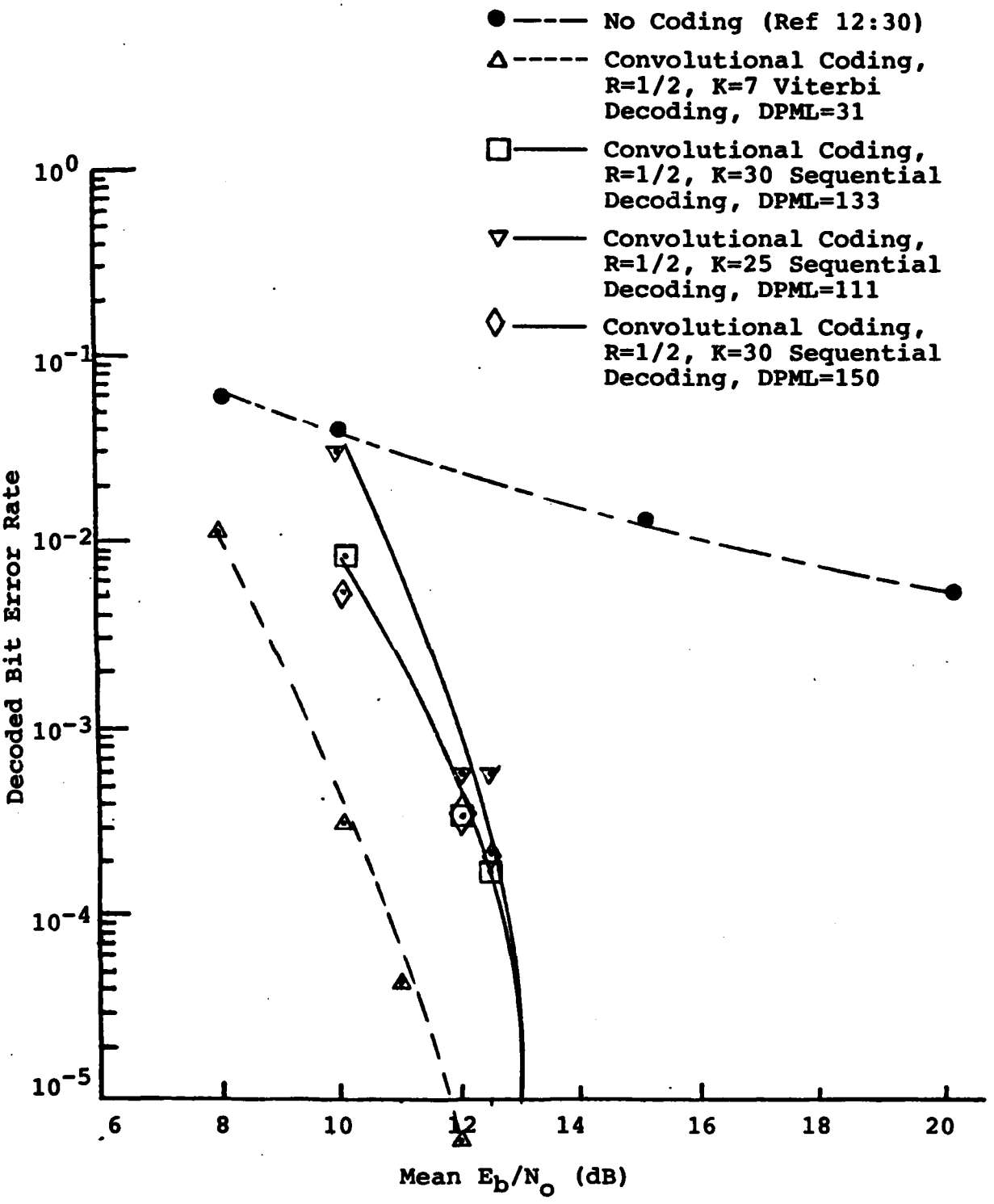


Figure 46. Encoded 2400 bps, DBPSK Link Performance in Moderately Slow Scintillation, τ_o = 0.05 sec.

TABLE XIX

Simulation Data for a 2400 bps, DBPSK Link in Moderately Slow Scintillation, τ_o = 0.05 sec

Mean E_b/N_o (dB)	Decoded Bits Total	Bit Error Rate	Ave. No. of Computations per Bit
Convolutional Coding,R=1/2,K=7-Viterbi Decoding,DPML=31			
8.0	60,120	1.100×10^{-2}	N/A
10.0	60,120	3.327×10^{-4}	N/A
11.0	300,240	4.663×10^{-5}	N/A
12.0	500,040	6.000×10^{-6}	N/A
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=133			
10.0	150,120	8.450×10^{-3}	4.243
12.0	150,120	3.531×10^{-4}	0.236
12.5	500,040	1.800×10^{-4}	0.496
13.0	500,040	0	0.1166
Convolutional Coding,R=1/2,K=25-Sequential Decoding,DPML=111			
10.0	150,138	3.097×10^{-2}	12.496
12.0	150,120	5.795×10^{-4}	0.597
12.5	500,040	5.840×10^{-4}	0.52
13.0	500,040	0	0.12
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=150			
10.0	150,120	5.030×10^{-3}	2.44
12.0	150,120	3.531×10^{-4}	0.24
12.5	500,040	2.220×10^{-4}	0.47
13.0	500,040	0	0.12

TABLE XX

Viterbi Decoded Performance Gain Over the Sequential Decoded Links in Moderately Slow Fading

BER	R = 1/2, K = 30 Code	R = 1/2, K = 25 Code
10^{-3}	2.0 dB	2.5 dB
10^{-5}	1.5 dB	1.5 dB

are significant for $E_b/N_o > 10$ dB . Finally, Table XX lists the approximate performance gain of the Viterbi decoded link over the sequential decoded systems.

Figure 47 and sumulation data results in Table XXI were obtained for $\tau_o = 0.16$ sec . This value of τ_o corresponds to relatively slow scintillation for the 2400 bps, DBPSK, coded links shown here. Recall for the slow fading channel, amplitude fading is the predominant contributing factor for demodulated symbol errors. Fewer errors are generated as a result of phase perturbations occuring in the channel. Because the DBPSK demodulated symbol errors occur in clumps more frequently, the synchronous deinterleaver has greater difficulty in randomizing the errors. Therefore, during slow amplitude fading periods, deinterleaved symbol errors inputted to the random error-correction decoders are no longer memoryless. As a result, Figure 47 indicates larger E_b/N_o's are required to obtain comparible bit error probability performance found in moderately fast and moderately slow fading channel conditions.

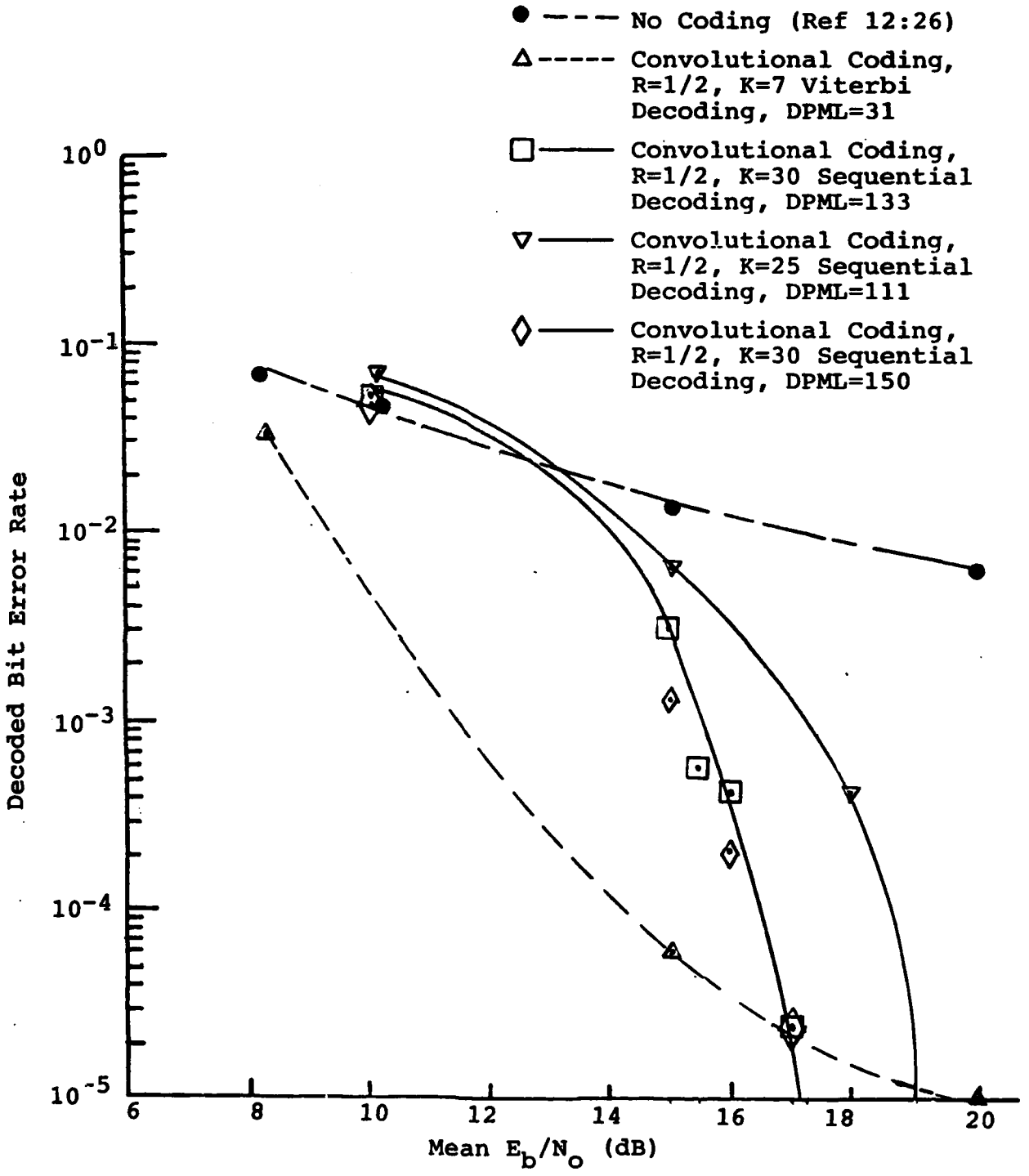


Figure 47. Encoded 2400 bps, DBPSK Link Performance in Slow Scintillation, τ_o = 0.16 sec .

TABLE XXI

Simulation Data for a 2400 bps, DBPSK Link in Slow Scintillation, τ_o = 0.16 sec

Mean E_b/N_o (dB)	Total Decoded Bits	Bit Error Rate	Ave. No. of Computations per Bit
Convolutional Coding,R=1/2,K=7-Viterbi Decoding,DPML=31			
8.0	120,240	3.409×10^{-2}	N/A
15.0	120,240	6.653×10^{-5}	N/A
20.0	500,040	9.999×10^{-6}	N/A
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=133			
10.0	100,080	5.280×10^{-2}	25.67
15.0	100,080	3.028×10^{-3}	1.264
15.5	500,040	6.080×10^{-4}	0.44
16.0	500,040	4.320×10^{-4}	0.529786
17.0	500,040	2.800×10^{-5}	0.22
20.0	100,080	0	0.07267
Convolutional Coding,R=1/2,K=25-Sequential Decoding,DPML=111			
10.0	100,080	7.334×10^{-2}	34.0
15.0	100,080	6.964×10^{-3}	2.955
18.0	500,040	4.450×10^{-4}	0.29
19.0	500,040	0	0.083
Convolutional Coding,R=1/2,K=30-Sequential Decoding,DPML=150			
10.0	100,080	5.043×10^{-2}	30.96
15.0	100,080	1.439×10^{-3}	0.6436
16.0	500,040	2.120×10^{-4}	0.30
17.0	500,040	2.800×10^{-5}	0.22

The most significant result illustrated in Figure 47 is the fact the Viterbi decoded link loses its advantage over the sequential decoded links for bit error rates $< 10^{-5}$. All three sequential decoded systems begin outperforming the Viterbi decoded system as they approach E_b/N_o values corresponding to their theoretical limits. As shown in the previous scintillated channels, the slope of the Viterbi bit error probability performance curve steepened as it approached E_b/N_o values corresponding to the sequential decoder's theoretical limits. In the slow fading channel, however, the curve's slope becomes more shallow. In constrast, the sequential decoded links maintain their theoretical limit performance characteristics. Clearly, the sequential decoders handle the slow amplitude fading channel conditions better than the Viterbi for bit error rates $< 10^{-5}$.

The $K = 30$ convolutionally encoded system approaches a theoretical limit of 17.25 dB and the $K = 25$ encoded link approaches a theoretical limit of 19 dB. Here again the $K = 30$ sequentially decoded link with DPML = 150 is essentially equivalent in performance to the link with DPML = 133. Lastly, the Viterbi vs sequential decoded link performance gain chart is provided in Table XXII.

Before summarizing the system implications based upon the scintillated channel results presented above, Figures 48 through 50 have been included to illustrate the characteristics of messages that were received in error. These messages

TABLE XXII

Viterbi Decoded Performance Gain of the R = 1/2, K = 7 Encoded Link Over the Sequential Decoded Links in Slow Fading

BER	R = 1/2, K = 30 Code	R = 1/2, K = 25 Code
10^{-3}	4.35 dB	5.9 dB
10^{-5}	-2.75 dB	-1.0 dB

were obtained from the link simulations performed for the moderately fast fading channel. However, the received message error characteristics shown in these figures are the same for the other scintillated channels as well. Figures 48, 49, and 50 again show Viterbi decoded bit errors produce a short sequence of character errors; whereas, the sequentially decoded errors occurred in widely separated burst, entailing a larger number of message characters.

Implementation Trade-Offs for System Application

Comments summarizing the performance and implementation trade-offs between the short constraint length encoded link with Viterbi decoding and the long constraint length encoded links with sequential decoding are presented for a digital satellite communication system that is expected to operate in a prolonged nuclear scintillated environment. To reiterate, the link under consideration incorporates rate 1/2 encoding of 2400 bps data and utilizes DBPSK modulation/demodulation.

As discussed in the preceding paragraphs, a 10^{-5} bit error probability was found as a useful lower bound for

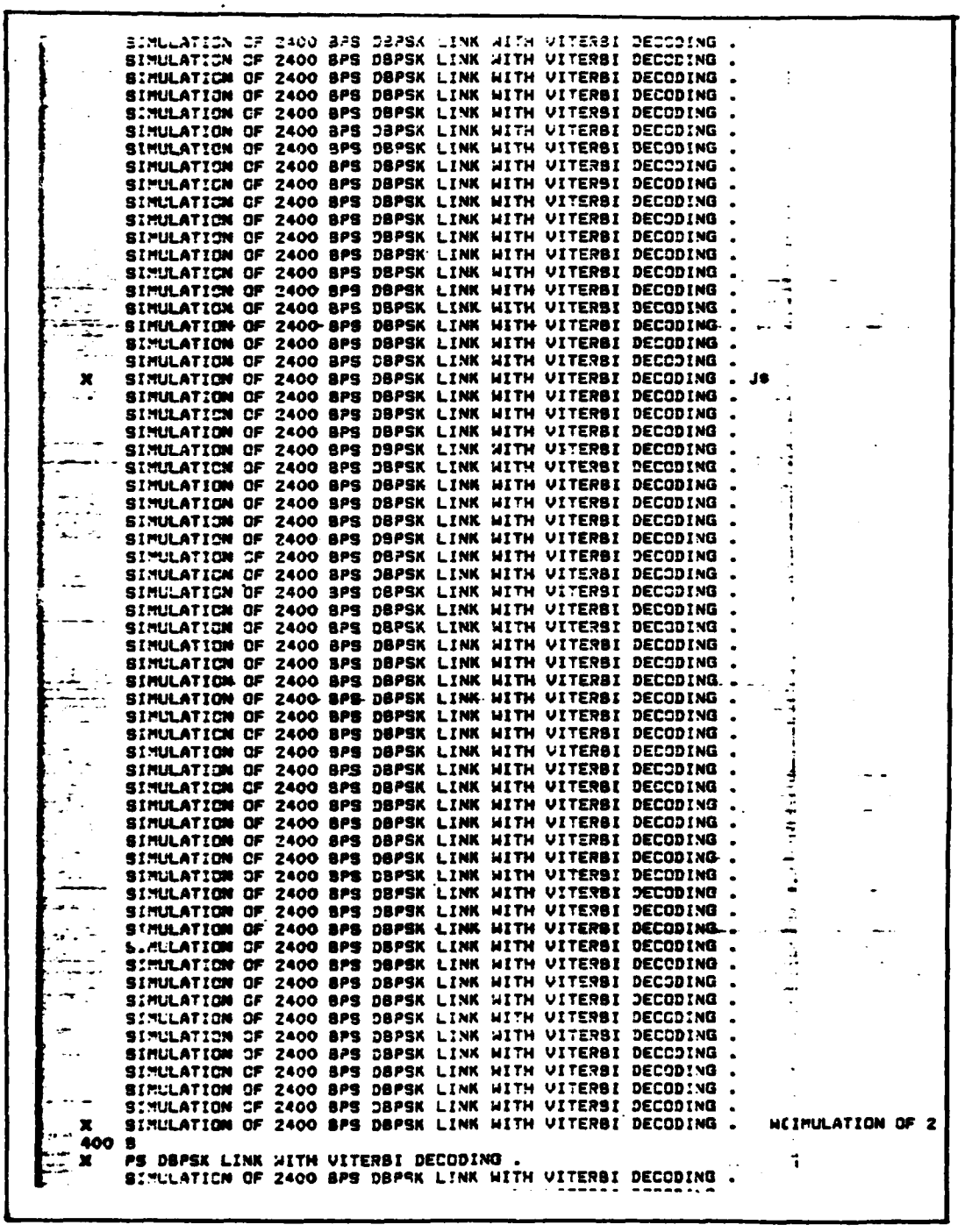


Figure 48. Viterbi Decoded Message Errors in Scintillated Signal Fading Conditions.

```
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMLLATIA^3XN(:3R06PZ^-5F4-C <G$N7UYXD$YLS$A
X    TKEH,DUW+L!VC3.<S!9<LHL<<<1
X    3!&T)OPR9S$CU32A(\KN:BVCC$<L&V!IO\GZDTU^BL=ABGO:$KYAV+S<I+C50D9
X    W!$5GR:P$PM2C04(N#:VYBZCW_AGG.TJA^(45ONUW&G(0*"%E7NS=A
     #,X3(A-$@ A\W$DG?S:B-K-(%U"\H(VX4UE.LK P'\A'
     U9NR
X    )RI_T8HF!$!$)GZZTZ0F
     G3G9R-CXFE:LKZ
X    I&!O#CXG'+<MG+&DWSE"OG9-\*Z3GIFJT/I^ )KZ00FD1PGCHDKTPK LINK MPGX SEZW(3.C<E3
<GZJT
X    GLED.S89TCEST:ZF9/>0:\ 1 VHG$UJ9G3NJNL4B
X    A4@\RE+TY$DIGP& U3G YJ<-$TVJMLLATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL
DECCD
X    ING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUE
```

Figure 49. Sequential Decoded Message Errors in Scintillated Signal Fading Conditions with K = 30 Code.

```
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
X    SIMULATION OF 2400 BPS DBPSK LINK WIZ3AYKP30/SIZ3:3C+LD;8C7.
X    3Y'EHS5YCO DO&C6304$DW@AF\ON4J_SI2>+$+)3/7M!/\CU +IZ#N3H^G,NCYMMJI!1I0.80.19
0<P(S
     R3!3ZOWK"))LI>C,R+LKLGUZ P(I):D5I
X    HES:#%M1)ED8#9I'N9D'=<&.4 9SC%0ADWK(D,_E"I UM3#]=-J+VX7L5-T3CA3%
X    )+:WH(SLS8GCP&IE$"%S1$$\)APR:3K LINK WITH S1X'529MAMP'ISJHMGB-D/39KW
X    S %3. K& +"3P *3R)S2UT+ L>*8 F*U (SU3P]H4!!J 3%CLLK.30+
     49.3ME3KK<!/& F5UP8H=RON3[RKON->JA&KU8'&1SSED$K#DE,GC
X    LMKF$5)CYM_-WR5\JTC^Z2C4Q:*@R ,S3SK+LKLI)VISH"35PLDN>O"$4EE"ILG/7$(YJL,\,PSI
KG L$
X    7TC$VXF(BA\H(KM FCH[97A_*$RES=E)PKR%D3EM9EV+V8^ID4JG3D^4-,] X<8%Z',X!PF#7H
0<((+A
X    X3JX<"Y.2WF,C9Z-(X=T_8XV&H'S)H J50LKC88(F332WP$8FX# FV23"D!\J6KXOX"Y=K'%NJ)8
E8/E/

X    3IH#FN&IK5E,A\3GE@+
X    "$&5.9:433P'E:HS!(QIHN,24;GIT:G&EZCO2EKIKNO.3UN-C3W3T TSGC3L8+ 4:440 3.DPY58
..ZPY
X    D:3H9X[EHNCXD)E3D,$3_F$LN8+3W+A5L35I[L3RS=+SV=4VZ[S483]C"M_'D+;8$IOPRD8MO_+C
A(#E#
X    NFYNKP)/SM"1HUS('^OO1::OO!$XAG8IOZCI<<A@="CG@X(SF_881\Y@/NDT'"DYKA!N&BYLS8%^
JHKDN
X    $"7\SJRM9)+DC9_K+K9X+ _J4$%PC87C84C-M,AICNWKLVX4>3LI!_OO+CNFJV T45"YROGA(O '
O>&F^
     G'YM=I1RD''-J"]$48A(H
X    5IJC"-45,]I\H2IWN$_3,15370^<G3FS1WSOHYNJ5^1%9 UW@4WV4-%-FVVSC"KJ^M9
     SKIK$8^57K5MF8S1'$(R:G\DW8R2"X
X    'N+FI ,+V+\MR^P+GMJLWJ"L)ACD>O^'AVTCTY/.5.$ 558!$IM99D98\COMJNHCM3PH,C)SUM>T
SASVT
X    J)_$+NA$'Y3,+TJXM#KK _GKSXO<(ZPNHG SPZLL)NO TE2ZOSKI$S4^)%J'@IJ'V&TDF+;7)U]\
)\AH.
X    :97H2::0 CR\$DISSA OC^R979:@=O\[5)O-I,]YFFCWF)+O H
X    ^M]4LSI^KK1DFP2HT 1HPV4F&#8K DHV:##YSNX3WO E:7DCD$$JY9+;-SO+<V(]G@\=J+FP^LP&
;A^A(
X    OK!B.;R83UD^^NGMT(IWFGUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
     SIMULATION OF 2400 BPS DBPSK LINK WITH SEQUENTIAL DECODING .
```

Figure 50. Sequential Decoded Message Errors in Scintillated Signal Fading Conditions with K = 25 Code.

TABLE XXIII

E_b/N_o Required to Maintain 10^{-5} Bit Error Rate as a Function of Scintillation Fading Rate

Decorrelation Time τ_o		
0.016 sec	0.05 sec	0.16 sec
R = 1/2, K = 7 Coding - Viterbi Decoding		
10.75 dB	11.5 dB	20.0 dB
R = 1/2, K = 30 Coding - Sequential Decoding		
11.5 dB	13.0 dB	17.25 dB
R = 1/2, K = 25 Coding - Sequential Decoding		
12.5 dB	13.0 dB	19.0 dB

characterizing link performance. In essence, this bound was superficially generated as a result of the link simulations performed for the sequential decoded systems. In particular, bit error rates $< 10^{-5}$ were not found for the sequentially decoded links due to the theoretical limits of performance and unreasonable computer simulation time required to detect lower error probability. In addition, a bit error probability of 5×10^{-5} was requested by the AFWL sponsor as a maximum lower bound for quantizing the performance.

Using the basis of comparison established above, Table XXIII lists the approximate E_b/N_o values required to obtain a bit error probability performance of 10^{-5} for the convolutionally encoded systems. Figures 51, 52, and 53 are also

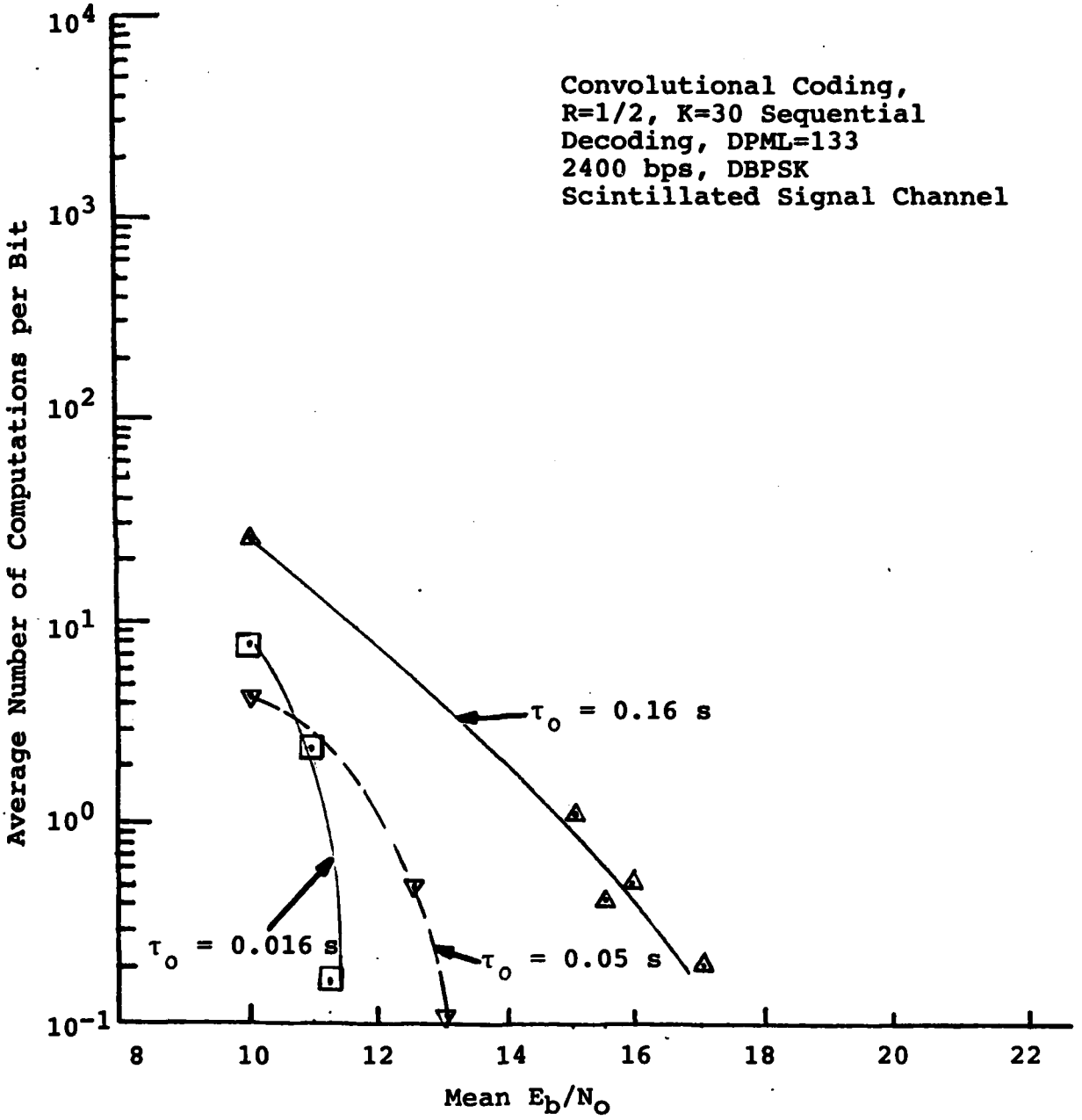


Figure 51. Average Number of Computations to Decode a Data Bit in a Scintillated Channel with R=1/2, K=30 Convolutional Encoding.

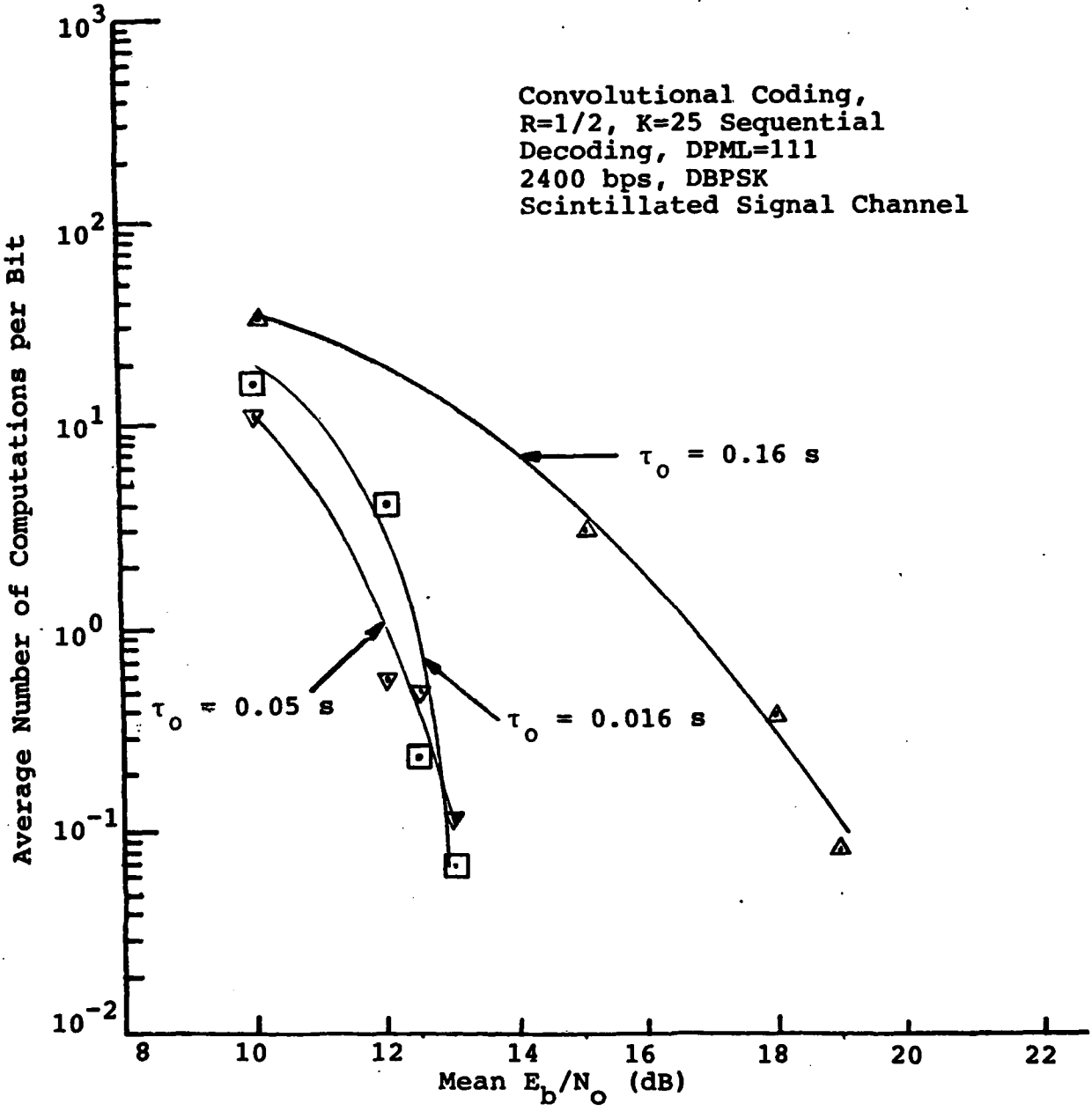


Figure 52. Average Number of Computations to Decode a Data Bit in a Scintillated Channel with R=1/2, K=25 Convolutional Encoding.

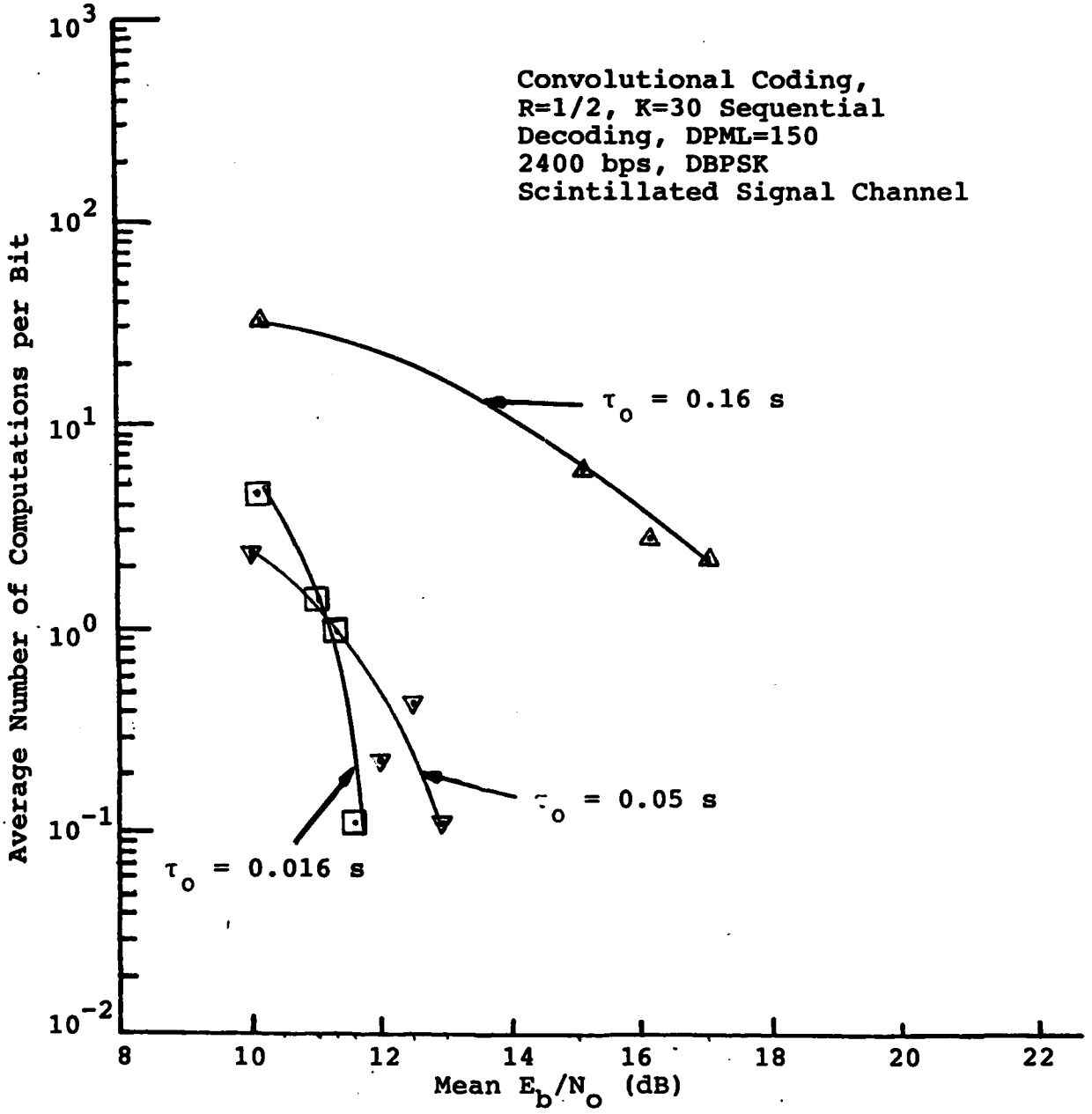


Figure 53. Average Number of Computations to Decode a Data Bit in a Scintillated Channel with R=1/2, K=30 Convolutional Encoding and DPML=150.

presented in this section to illustrate the average number of computations per bit required by each sequential decoder to produce a final data bit decision. This measurement is provided for the three scintillated channels simulated in this investigation. As discussed in the AWGN section, the computational values provide a rough measure of the decoder bit processing time delay. Hence, time delays become a major consideration of implementation trade-offs between the Viterbi and sequential decoded systems.

Based upon the simulation results for the convolutionally encoded systems in moderately fast, moderately slow, and slow fading conditions, a satellite communications link requiring a bit error rate $10^{-3} \geq \text{BER} \geq 10^{-5}$ for a given E_b/N_o less than the theoretical limit of any sequential decoded system would obtain maximum performance using the Viterbi decoded link configuration. This conclusion is based upon the rather clear-cut performance gain of the Viterbi decoded short constraint length code over the sequentially decoded long constraint length codes shown in Figures 45,46, and 47. In addition, Figures 51, 52, and 53 illustrate the additional time delay induced by each of the sequential decoders to produce a final decoded bit decision at E_b/N_o values less than their theoretical limit. The extra time here needed to search through the code tree can also induce further degradation in performance due to overflow. Therefore, the total average processing time given by Eq (53) is a major

contributing factor in selecting the most appropriate coding configuration.

Computational complexity is not a major contributing factor in terms of implementation trade-offs in this investigation. Computational complexity of Viterbi decoding increases exponentially with the constraint length, but the short constraint length 7 code used with Viterbi decoding is very practically implemented. In addition, the Viterbi decoded system parameters were not varied because they function as a basis for comparison with the other sequential decoded systems. Unlike the Viterbi decoder, sequential decoder complexity doesn't substantially increase with constraint length. However, it should be mentioned 3-bit soft-decision sequential decoding as implemented in this analysis, is more difficult to implement than hard-decision decoding (Ref 18:306).

The selection of the Viterbi decoded link versus the sequential decoded link for a satellite communication system requiring a bit error probability $< 10^{-5}$ is not as "clear-cut". Several considerations based upon the simulation results above must be addressed.

First, the trends of bit error rate vs E_b/N_o performance curves in Figures 45 and 46 show the difference in E_b/N_o required to obtain some bit error probability $< 10^{-5}$ becomes increasingly small. Thus, time delays associated with decoding a single bit may become the determining factor. Even then, however, Figures 51 and 52 illustrate sequential

decoder computational requirements get progressionally smaller as E_b/N_o increases.

For example, averaging the number of computations per bit for E_b/N_o values near the theoretical limit of the sequential decoders in Figures 51 and 52, one obtains approximately 0.1 computations per bit. Using this value and Eqs (42) and (53), the total bit processing time delay is given by

$$T = T_D + (1.1)(0.1)T_H \qquad (56)$$

$$= T_D + .11T_H$$

where

T_D = time delay associated with the decoded path memory length

T_H = cycle time required to complete one search in the sequential decoder

Given the hardware/software decoder has a search time, $T_H = 10$ ms , the $K = 30$ sequential decoded system has a total average time delay

$$T = 0.055 + .11(.010)$$

$$= 0.056 \text{ sec}$$

and the $K = 25$ sequential decoded system has a total average time delay

$$T = 0.046 + .11(.010)$$

$$= 0.047 \text{ sec}$$

Comparing these two values with the Viterbi decoder time delay, $T = 0.013$ sec , the bit processing time delay of the sequential decoders becomes less significant.

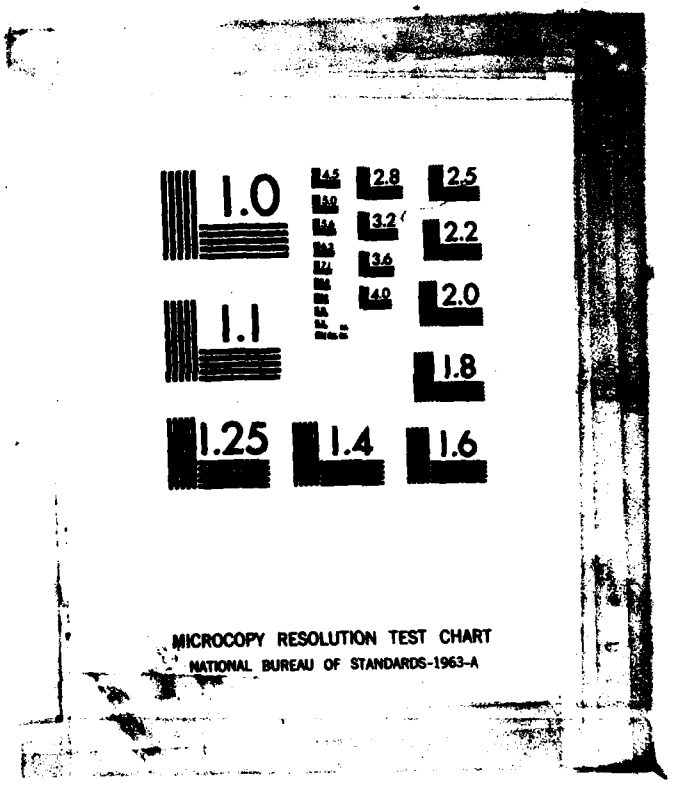
The third consideration, and possibly the most important, is the performance gain of the sequential decoded link over the Viterbi decoded link in the slow amplitude fading channel illustrated in Figure 47. To obtain a bit error probability $< 10^{-5}$ the Viterbi decoder performance curve indicates much larger values of E_b/N_o are required compared to the steep sequential decoder performance curves.

Fourth, Figure 4 in Chapter II shows slow fading is the dominant degradation effect for EHF links, at least in terms of time duration. For example, a scintillated channel operating at 20 GHz with a signal decorrelation time, $\tau_o = 0.16$ sec , can degrade link performance approximately 25 minutes after a single detonation. But a moderately slow fading channel with $\tau_o = 0.05$ sec and a moderately fast fading channel with $\tau_o = 0.016$ sec will provide moderate to severe phase and amplitude fading lasting 8 minutes and 2.5 minutes respectively. Even slower fading channels ($.16 < \tau_o < 0.9$ sec) are potential threats to EHF links and can last as long as one hour after a nuclear detonation (Ref 12:8-12).

Based upon the considerations above, the sequential decoded links evaluated in this analysis become more competitive with Viterbi decoding for links requiring bit error

AD-A124 814 PERFORMANCE OF SEQUENTIALLY DECODED LONG CONSTRAINT LENGTH CODES FOR A SA.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. J A FRAZIER 3/3

UNCLASSIFIED DEC 82 AFIT/GE/EE/82D-32 F/G 17/2.1 NL


1.0
1.1
1.25
1.4
1.6
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

rates $< 10^{-5}$. As a result, choices for either system may depend upon ease of implementation of 3-bit soft-decision sequential decoding, potential overflow problems, ease of decoded bit synchronization, and message error rate characteristics.

Chapter IX. Conclusions and Recommendations

Conclusions

The goal of this project was to determine the performance and implementation trade-offs between a short convolutional code using Viterbi decoding and long convolutional codes with sequential decoding in undisturbed and signal scintillation fading conditions. A major part of this task involved the development of a sequential decoder subroutine for the AFWL FSK-PSK Link Performance Code. Using this code, performance results for the two convolutional encoded systems were generated through detailed digital simulations of a 2400 bps, differentially coherent binary phase-shift-keyed satellite communications link.

The comparative analysis between the channel encoded systems was based upon the implementation of a rate 1/2, constraint length 7 code with Viterbi decoding; and two long, rate 1/2 codes with constraint lengths 25 and 30 using sequential decoding. Performance results were obtained in terms of bit error probability versus received bit energy-to-noise density ratio. In addition, implementation trade-offs between the Viterbi and sequential decoded systems were analyzed in terms of potential decoder induced time delays and relative performance gain.

The results obtained for the Viterbi decoded and sequential decoded link performance assessment are stated as follows:

1. In general, link simulation performance results showed long constraint length codes (K= 25 to 30) with sequential decoding did not provide any significant performance improvement over the short constraint length code with Viterbi decoding.
2. The 2400 bps, DBPSK satellite link with Viterbi decoding outperformed the sequentially decoded links in the additive white Gaussian noise channel for bit error probabilities greater than 10^{-5}.
3. In the moderately fast and moderately slow scintillated signal fading channels, the Viterbi decoded link again outperformed the sequential decoded systems for bit error probabilities greater than 10^{-5}.
4. In slow scintillated signal fading channels the sequential decoded links were outperformed by the Viterbi decoded systems for bit error rates greater than 2.5×10^{-5}. However, as the sequential decoded links approached their theoretical limits of performance the Viterbi decoded link advantage went away. Results showed the sequential decoded links outperformed the Viterbi decoded link for bit error probabilities less than 10^{-5} in slow fading conditions.
5. Sequential decoder computations required to decode a single bit of data caused longer decoder bit processing time delays in comparison to the Viterbi decoded link for bit error rates greater than 10^{-5}.

This was the case for both the undisturbed and scintillated channels.

Recommendations

Several follow-on efforts became evident as a result of the work performed during the period of this thesis. Potential future projects using the FSK-PSK Link Performance Code are listed below.

1. Investigation of sequential decoder performance improvement when design parameter values, such as, the initial threshold, threshold increment, and integer symbol metrics are varied.
2. Analysis of performance and implementation tradeoffs of rate 1/3 short constaint length codes with Viterbi decoding and rate 1/3 long constraint length codes with sequential decoding. This project would also involve a search for good rate 1/3 convolutional codes.
3. Characterization of sequential decoder performance for other convolutional codes. This project would entail developing a data base on the performance of sequential decoded systems for various convolutional codes with rates 1/2 - 1/8 and constraint lengths 10 - 32. This assessment could be performed for the additive white Gaussian noise and scintillation channels.

4. Examination of sequential decoder performance when the decoder is forced to make an output bit decision after so many bit computations have taken place. Here, the investigator would be required to make slight modification to the current sequential decoder subroutine.

5. Performance assessment of sequentially decoded links in extremely slow fading conditions for signal decorrleation times between 0.5 and 1.0 seconds. A major part of this project would involve finding an optimum synchronous interleaver in the sense of maximum randomization of burst errors with minimal interleaving/deinterleaving induced time delay.

6. Quantification of Viterbi and sequential decoded message error rate performance for the undisturbed and scintillated channels . Message error probabilities are particularly important to satellite communication systems which require minimal message erasures. In several of the link simulations performed in this thesis, the sequential decoded message error rate performance was competitive with the Viterbi decoded message error rate performance.

Bibliography

1. Barrows, J. T. and M. Leiter. Sequential Decoder Simulation Study. MTP-106. Bedford, Massachusetts: Mitre Corporation, January 1970.

2. Blustein, G. and K. L. Jordan, Jr. An Investigation of the Fano Sequential Decoding Algorithm by Computer Simulation. Group Report 62-G. Lexington, Massachusetts: M. I. T. Lincoln Laboratory, July 1963.

3. Bogusch, R. Digital Simulation of PSK Modems. MRC-R-333. Santa Barbara, California: Mission Research Corporation, December 1977.

4. Bogusch, R. L. and F. W. Guigliano. Design and Evaluation of Survivable Links for IONDS. MRC-R-598. Santa Barbara, California: Mission Research Corporation, December 1980.

5. Bogusch, R. L. and D. L. Knepp. Propagation Effects on GPS Receiver Model X. AFWL-TR-79-25. Kirtland AFB, New Mexico: Air Force Weapons Laboratory, September 1979.

6. Bogusch, R. L. Preliminary Analysis of Propagation Effects on GPS. AFWL-TR-78-126. Kirtland AFB, New Mexico: Air Force Weapons Laboratory, January 1977.

7. Bogusch, Robert L. , et al. Signal Propagation Effects on Selected Satellite Systems. AFWL-TR-78-170. Kirtland AFB, New Mexico: Air Force Weapons Laboratory, April 1979.

8. Costello, Daniel J., Jr. and James L. Massey. "Non-systematic Convolutional Codes for Sequential Decoding in Space Applications," IEEE Transactions on Communication Technology, COM-19 (5): 806-813 (October 1971).

9. Fano, Robert M. "A Heuristic Discussion of Probabilistic Decoding," IEEE Transactions on Information Theory, IT-9: 64-74 (April 1963).

10. Forney, G. David, Jr. "Coding and its Application in Space Communications," IEEE Spectrum: 47-58 (June 1970).

11. Forney, G. David, "The Viterbi Algorithm," Proceedings, 61 (3): 268-278 (March 1973).

12. Frazier, James A., Jr. and Gary S. Krajci. The Performance of an EHF Satellite Communications Link in a Scintillated Environment. AFWL-TR-81-33. Kirtland AFB, New Mexico: Air Force Weapons Laboratory, July 1981.

13. Frazier, James A., Jr. Nuclear Effects on the Space Based Radar System. AFWL-TN- NTYCC-81-6. Kirtland AFB, New Mexico: Air Force Weapons Laboratory, March 1981.

14. Gallager, Robert G. Information Theory and Reliable Communication. New York. John Wiley and Sons, 1968.

15. Geist, John M. "An Empirical Comparison of Two Sequential Decoding Algorithms," IEEE Transactions on Communication Technology, COM-19 (4): 415-419 (August 1971).

16. Heller, Jerrold A. and Irwin Mark Jacobs. "Viterbi Decoding for Satellite and Space Communication," IEEE Transactions on Communication Technology, COM-19 (5): 835-848 (October 1971).

17. Ibanaki, Ronald., et al. CPSK, DPSK, and FSK Demodulator Performance Under Nuclear Stressed Conditions. DNA 5006F. Washington D.C.: Defense Nuclear Agency, March 1979.

18. Jacobs, Irwin M. "Practical Application of Coding," IEEE Transactions on Information Theory, IT-20 (3): 305-310 (May 1974).

19. Jacobs, Irwin Mark. "Sequential Decoding for Efficient Communication from Deep Space," IEEE Transactions on Communication Technology, COM-15 (4): 492-501 (August 1967).

20. Jordan, K. L., Jr. "Performance of Sequential Decoding in Conjunction with Efficient Modulation," IEEE Transactions on Communication Technology, COM-14: 283-297 (June 1966).

21. Larsen, K. J. "Short Convolutional Codes With Maximal Free Distance for Rates 1/2, 1/3, and 1/4," IEEE Transactions on Information Theory, IT-19: 371-372 (May 1973).

22. Michelet, Allen H. A Computer Simulation of Maximum Likelihood Decoding of Convolutionally Encoded Data Propagated Through a Disturbed Medium. DNA 4342T. Washington D.C.: Defense Nuclear Agency, September 1977.

23. Odenwalder, Joseph P. Error Control Coding Handbook. San Diego, California: Linkabit Corporation, July 1976.

24. Ramsey, John L. "Realization of Optimum Interleavers," IEEE Transactions on Information Theory, IT-16 (3): 338-345 (May 1970).

25. Viterbi, Andrew J. "Convolutional Codes and Their Performance in Communication Systems," IEEE Transactions on Communication Technology, COM-19 (5): 751-772 (October 1971).

26. Viterbi, Andrew J. and Jim K. Omura. Principles of Digital Communication and Coding. New York: McGraw-Hill Book Company, 1979.

27. Wiggert, Djimitri. Error Control Coding and Applications. Bedham, Massachusetts: Artech House Incorporated, 1978.

28. Wittwer, Leon A., "A Trans-Ionospheric Signal Specification For Satellite C^3 Applications," Robust Communication Links of WESCON 80 Session Five. Anaheim, California, September 16-18, 1980.

29. Wozencraft, John M. and Irwin Mark Jacobs. Principles of Communication Engineering. New York: John Wiley and Sons, 1965.

Appendix A: FSK-PSK Link Performance Code User's Manual

Overview of FSK-PSK Code

The FSK-PSK Link Performance Code was designed, coded, and run on a Digital Vax 11/780 using VMS compiler version 2.5 and 3.0. Code is written in Fortran 77, and consist of two main programs along with several subroutines for simulation of a digital satellite communications link.

Generation of the Data File Overview

1. Data file is set up in a standard card format, 80 characters long. Refer to Figures A.1 and A.2.
2. All input parameters, option numbers, and etc., must be typed all the way to the right of the designated space except in the case where a decimal point is used in the numerical expression.
3. Data File contains the receiver design parameters, and received signal parameters to be used as input data for any given link simulation.
4. Card or row number must be typed into the first column of each data file row. This row number identifies the kinds of parameters to be inputted into the program. Refer to Figures A.1 and A.2.
5. Although data files are typically generated on an interactive terminal, data file rows will be referred to as cards.

PSK Data File

Card #1, Simulation Options -1

Columns 2-10: Case Number

Function: Any convenient number to identify the output. Useful if user intends to run several cases. (-) in front of case number implies the next case will use the same message text found on the following card.

Columns 11-20: Trial Selection

Function: For a normal job type 0 in column 20. (-n) is used when user desires to change the random number seed value. (n) is the new random number value. In general, it is better to use 0 here.

Columns 21-30: Print Option

Function: A 0 is normally used here. By typing a 0 in column 30 the output will inform the user when the demodulator loses lock of the signal being simulated. That is, it tells what the demodulator did during any part of the simulation. It is suggested a 0 be used here.

A (-n) turns off the message indicating that the AFC, PLL or anything else has lost lock in the demodulation process. This option is not usually used.

A (+n) provides extra detailed information on the demodulator operation.

Columns 31-40: Number of Text Characters (1-80)

Function: Specifies the number of text characters in one line of message text. Title or message text can contain a maximum of 80 characters. Each character is interpreted in the simulation as a 6-bit ASCII Character.

Spaces are considered characters.

A negative sign (-) in front of this number prevents received messages from being printed onto the output file. This option becomes useful when the user is limited in file space on the disk and wishes to keep his output files down to a minimum.

Printing the messages, however, is very useful for analyzing the nature of resultant, bit, character, and message errors.

Columns 41-50 and 51-60: Random Number Seed

Function: Random numbers are needed for producing a noise sequence. Using the same random numbers for each case provides control over what noise sequence is used. Any two random numbers can be selected. They must be interger numbers.

Columns 61-70: Simulation Length (sec)

Function: This quantity is the desired message transmission time. Link simulation times for the AWGN channel are typically determined by the number of bits, frames, message lines or characters which will provide appropriate error statistics. In scintillated channel conditions the simulation time is based upon a necessary condition to process 300 to 500 decorrelation times, τ_o .

Considerations: Key considerations for determining the link simulation transmission time:

1. 6 bits/character
2. Time to transmit one message,

$$T_m = \frac{\text{(No. of Message Text Characters)(6 bits/Character)}}{\text{Data Rate}}$$

3. When interleaving and coding is used,

$$\text{Simulation Time} \cong \text{Interleaving Time Delay+Decoder Time Delay} + \text{(No. of messages desired)}T_m$$

Card #2

Columns 1-80: Title/Message Text

Function: An English language message is typically used because a greater variety of characters better demonstate link performance. However, the message text can utilize any desired character arrangement.

Card #3, Modulation/Interleaving -2

Columns 2-10: Data Bit Rate (bps)

Columns 11-20: Sampling Rate (Hz)

Function: At a minimum, the Nyquist sampling rate should be used. That is, a minimum of 2 times the data bit rate for uncoded links or 2 times the encoded symbol rate for convolutionally encoded links. On the link simulation output results this value is referred to as the A/D sampling rate.

Columns 21-30: Bits per Frame (bits)

Function: This value enables error statistics to be listed in the output in terms of frame error rate, block error rates, packet error rates, and so on.

Columns 31-40: Carrier Frequency

Function: Any carrier frequency (UHF-EHF) can be specified in Hz.

Columns 41-50: Differential Encoding/Decoding

Function: A value of 0 implies no differential encoding of data is desired. A value of 1 implies the user wishes to implement differential encoding for coherent phase-shift-keyed modems, i.e., BPSK, QPSK, and offset QPSK. With this scheme a differential decoder is implemented directly after the Viterbi or sequential decoder. A value of 2 is used for normal differentially coherent phase-shift-keyed modems, e.g., DBPSK and DQPSK.

Columns 51-60 and 61-70: Synchronous Interleaver Parameters

Function: N_2 is chosen with respect to the maximum fade duration the user wants to interleave over. N_1 is chosen with respect to the decoder path memory length. A detailed discussion of how these parameters are selected is given in Chapter VII. In general, an optimum synchronous interleaver can be implemented by selecting N_2 and N_1 to satisfy the following:

$$N_1 > 4\frac{K}{R} \quad , \qquad \text{(A-1)}$$

where the factor 4 is the minimum number of code constraint lengths over which errors should be uncorrelated, R is the code rate and K is the code constraint length.

$$N_2 \gtrsim (R_D/R)\tau_o \quad , \qquad \text{(A-2)}$$

where R_D is the uncoded data rate and τ_o is the signal decorrelation time or inverse of the signal fading rate. In addition, $N_2 > 2N_1$ is required and N_1 and N_2 must be relatively prime (N_1 and N_2 must not contain any common factors). The output will indicate if N_1 and N_2 aren't relatively prime. Common values for N_2 and N_1 are given below as

$$N_2 = 133$$
$$N_1 = 61$$

$$N_2 = 399$$
$$N_1 = 61$$

when $N_1 = 61$, largest value of N_2 can be is 900. If $N_1 = 59$, however, N_2 can be as large as 931. Interleaving can be disabled by setting

$$N_2 = N_1 = 1 \quad .$$

Card #4, Coding Parameters -3

Columns 2-10: Modulation Type

Columns 11-20: Convolutional Code Rate

Function: Convolutional code rate, R , is selected by inserting the number of desired modulo-two adders, where R = 1/(No. of Mod-2 Adders) .

The following code rates can be implemented: 1/2, 1/3, 1/4, 1/5, 1/6, 1/7, and 1/8. To disable coding, set the number of modulo-two adders equal to 1.

Columns 21-30: Convolutional Code Constraint Length

Function: Typical convolutional code constraint lengths for Viterbi decoded link simulations are, K = 3 to 8 . Sequential decoded links use code constraint lengths, K = 10 to 32 .

Columns 31-40, 41-50, and on

Card #5, Columns 1-10, 11-20, 21-30, 31-40, 41-50, and 51-60: Modulo-Two Adder Connection Patterns

Function: Modulo-two adder connections must be provided in decimal. Link simulation outputs provide the modulo-two adder connections in octal.

Card #4

Columns 51-60: Number of Quantization Bits

Function: Number of quantization bits can be specified for soft-decision decoding.

1 - Hard decisions
2 - 4 level quantization
3 - 8 level quantization

Columns 61-70: Decoder Path Memory Length (bits)

Function: Decoder path memory length specifies the number of decoded bits a decoder must process before an output bit decision is made.

Card #6, AGC Parameters -4

Columns 2-10: AGC Time Constant

Typical Values: 1.0 sec and 10.0 sec

Columns 11-30: Leave Blank

Columns 31-40: OQPSK Lock Detection Time Constant (sec)

Columns 41-50: OQPSK Lock Detection Threshold

Disable Function: When not using OQPSK, set lock detection time constant and threshold to 0.0 .

Columns 51-60: Leave Blank

Columns 61-70: AGC Type

Types: 1 = coherent
2 = envelope
3 = square-law

Card #7, PLL Parameters -5

Columns 2-10: PLL Bandwidth (Hz)

Typical Values: 21 Hz, 175 Hz

Columns 11-20: PLL Damping Factor

Typical Values: 1-For 2nd order PLL only.

Columns 21-30: PLL Order

Permitted Values: 1, 2, or 3

Columns 31-40: PLL Lock Detection Time (sec)

Typical Values: .025 sec with PLL bandwidth = 200 Hz
.25 sec with PLL bandwidth = 20 Hz
1 sec with lower PLL bandwidths

Columns 41-50: PLL Lock Detection Threshold

Typical Values: .1 - .25

Columns 51-60: Leave Blank

Columns 61-70: PLL Type

Types: Costas and Power-Law

Card #8, AFC Parameters -6

Columns 2-10: AFC Loop Bandwidth (Hz)

Typical Value: 3.0 Hz

Columns 11-20: AFC Damping Factor

Typical Value: 1.0 Hz

Columns 21-30: AFC Order

Permitted Values: 1 or 2

Columns 31-40: AFC Lock Detection Time Constant (sec)

Typical Value: 1.0 sec

Columns 41-50: AFC Lock Detection Threshold

Typical Value: 0.15

Columns 51-60: Leave Blank

Columns 61-70: AFC Switch

Function Of Values: 0 - PLL detection circuit turns AFC on when phase lock is lost, i.e., AFC is operated automatically by PLL.

-1 - AFC is turned off
+1 - AFC is turned on (used with noncoherent demodulation)

Card #9, Signal Dynamics Parameter -8

Columns 2-10: Mean E_b/N_o (dB)

Function: Desired bit energy-to-noise density ratio for a given link simulation.

Columns 11-20: Total Electron Content (el/m^2)

Function: Total electron content the simulated signal would propagate through to induce signal absorption and attenuation effects. To disable type 0 .

Columns 21-30: Phase shift

Function: Initial phase shift or error between the receiver's local oscillator and the received signal. To disable type 0 .

Columns 31-40: Frequency Shift (rad/sec)

Function: Initial frequency error between receiver's local oscillator and the received signal. To disable type 0 .

Columns 41-50: Frequency Rate (rad/sec)

Function: Rate at which the frequency is changing as might occur from doppler effects. To disable type 0 .

Columns 51-60: Jerk (rad/sec^3)

Function: Rate at which the frequency-rate-of-change, is occuring. Jerk is typically associated with a changing doppler rate due to vehicle acceleration. To disable type 0 .

Columns 61-70: Jerk Duration (sec)

Function: Time duration of vehicle acceleration. To disable type 0 .

Card #10, Signal Propagation -9

Columns 2-10: Scintillated or AWGN Channel Selection

Function: For an undisturbed or additive white Gaussian noise channel type 0 . If a scintillated signal channel is desired type +1 .

Columns 11-20: Multiple-Phase-Screen Realization No.

Function: Leave blank if evaluating an AWGN channel. If a scintillated channel is simulated always type 1 .

Columns 21-30: MPS Offset (sec)

Function: Leave blank if simulating AWGN channel. This value allows this simulation to start at any point in the MPS realization. Typical value is 0 .

Columns 31-40: MPS Time Interval (sec)

Function: When using the MPS signal realization, the value of τ_o is determined by this value of MPS Δt (time interval) as follows:

$$\Delta t = \left[\frac{\Delta x}{l_o} \right] \tau_o$$

where Δx is the MPS grid resolution, and l_o is the signal decorrelation distance measured in the MPS realization. Values of Δx and l_o for the two realizations (data files) for this code are shown below.

1. MPS Realization Case 8091:
   l_o = 113.6 meters
   Δx = 1.83105 meters
2. MPS Realization Case 8143:
   l_o = 5.74 meters
   Δx = 1.83105 meters

Columns 41-70: Leave Blank

Card #11, MPS File Name

Function: Do not include this card if AWGN channel is simulated.

Columns 1-26: 8091POW.REL or
8143POW.REL

Card #12, Termination of Input Data

Function: A zero or blank in column 1 of this card terminates input data for each case.

FSK Data File

Card #1, same as above.

Card #2, same as above.

Card #3, same as above.

Card #4, same as above.

Card #5, same as above.

Card #6, AGC Parameters -4

Columns 2-10: AGC Charge Time Constant (sec)

Typical Value: 1.0 sec. To disable (turn AGC off) type a very large number like 1×10^{30} .

Columns 11-20: AGC Discharge Time Constant (sec)

Typical Value: 1.0 sec. To disable type a very large number.

Columns 21-30: AGC Maximum Voltage Gain

Typical Value: 50

Columns 31-40: AGC Minimum Voltage Gain

Typical Value: 0.02

Columns 41-50: AGC Feedback Gain Factor

Typical Value: 40

Card #7, same as Card #8 above.

Card #8, same as Card #9 above.

Card #9, same as Card #10 above.

Card #10, same as Card #11 above.

Card #11, same as Card #12 above.

Performing a Link Simulation on Digital VAX 11/780 Using VMS Compiler

1. Compile all FORTRAN Routines except the main PSK and FSK programs using the command 'FORTRAN'

2. PSK Link Simulation with Viterbi Decoding
   a. Generate PSK Data File, i.e., PSK.DAT; (some version #)
   b. FORTRAN PSK.FOR;1
   c. LINK PSK, CPSK, DPSK, CGAUSS, DCODE1, ERFC, SHUFL3, SHUFL4, SIGFSL, SIGMDL, SIGMPS, SIGNAL
   d. RUN PSK
   e. After simulation is completed, Type PSK.OUT; (some version #)

3. PSK Link Simulation with Sequential Decoding
   a. Generate PSK.DAT;#
   b. FORTRAN PSK.FOR;2
   c. LINK PSK, CPSK, DPSK, CGAUSS, DCODE2, ERFC, SHUFL3, SHUFL4, SIGFSL, SIGMDL, SIGMPS, SIGNAL
   d. RUN PSK
   e. TYPE PSK.OUT;#

4. FSK Link Simulation with Viterbi Decoding
   a. Generate FSK.DAT;#
   b. FORTRAN FSK.FOR;1
   c. LINK FSK, MFSK, CGAUSS, DCODE1, SHUFL3, SHUFL4, SIGFSL, SIGMDL. SIGMPS, SIGNAL
   d. RUN FSK
   e. TYPE FSK.OUT;#

5. FSK Link Simulation with Sequential Decoding
   a. Generate FSK.DAT;#
   b. FORTRAN FSK.FOR;2
   c. LINK FSK, MFSK, CGAUSS, DCODE2, SHUFL3, SHUFL4, SIGFSL, SIGMDL, SIGMPS, SIGNAL
   d. RUN FSK
   e. TYPE FSK.OUT;#

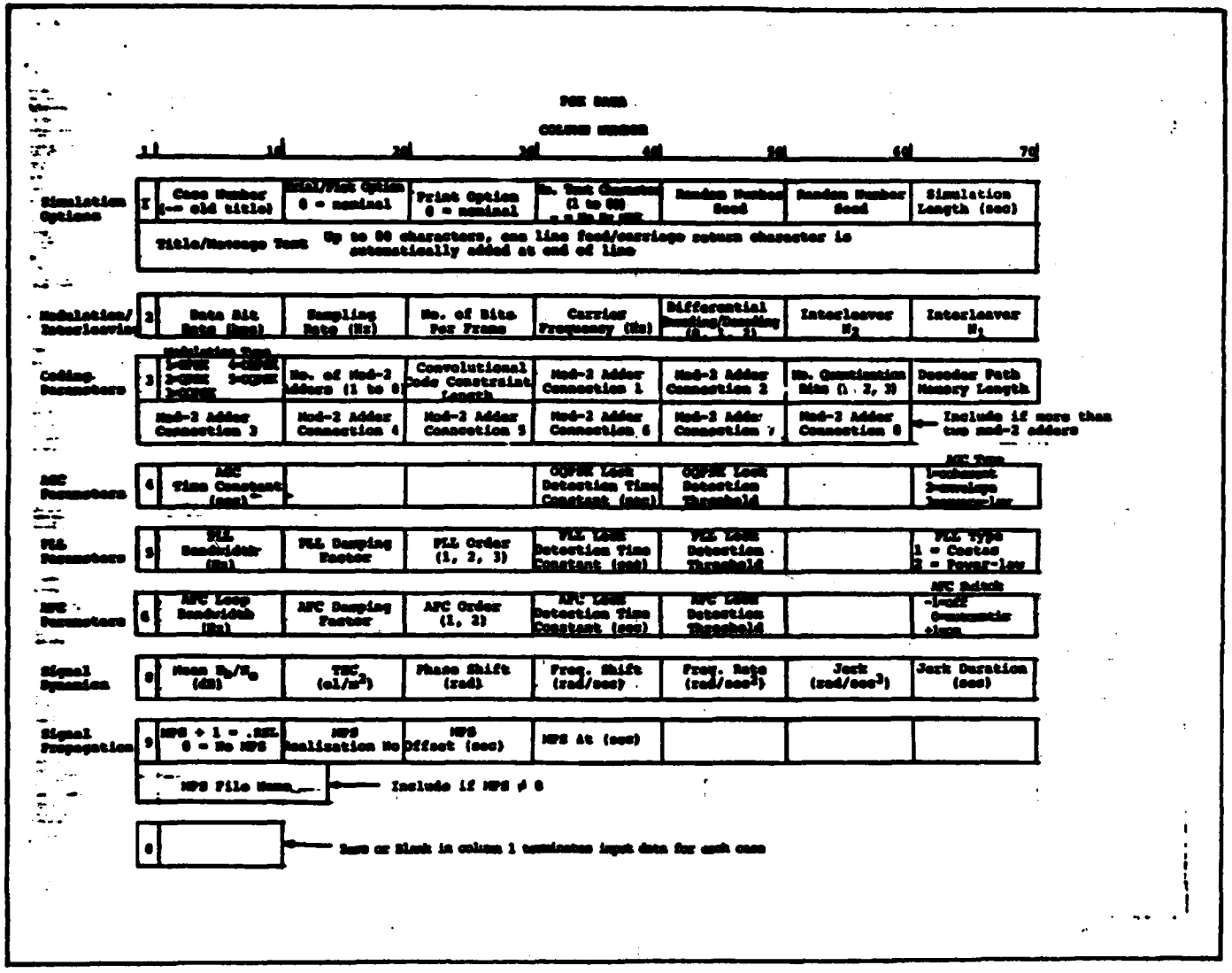


Figure A.1. PSK Data File

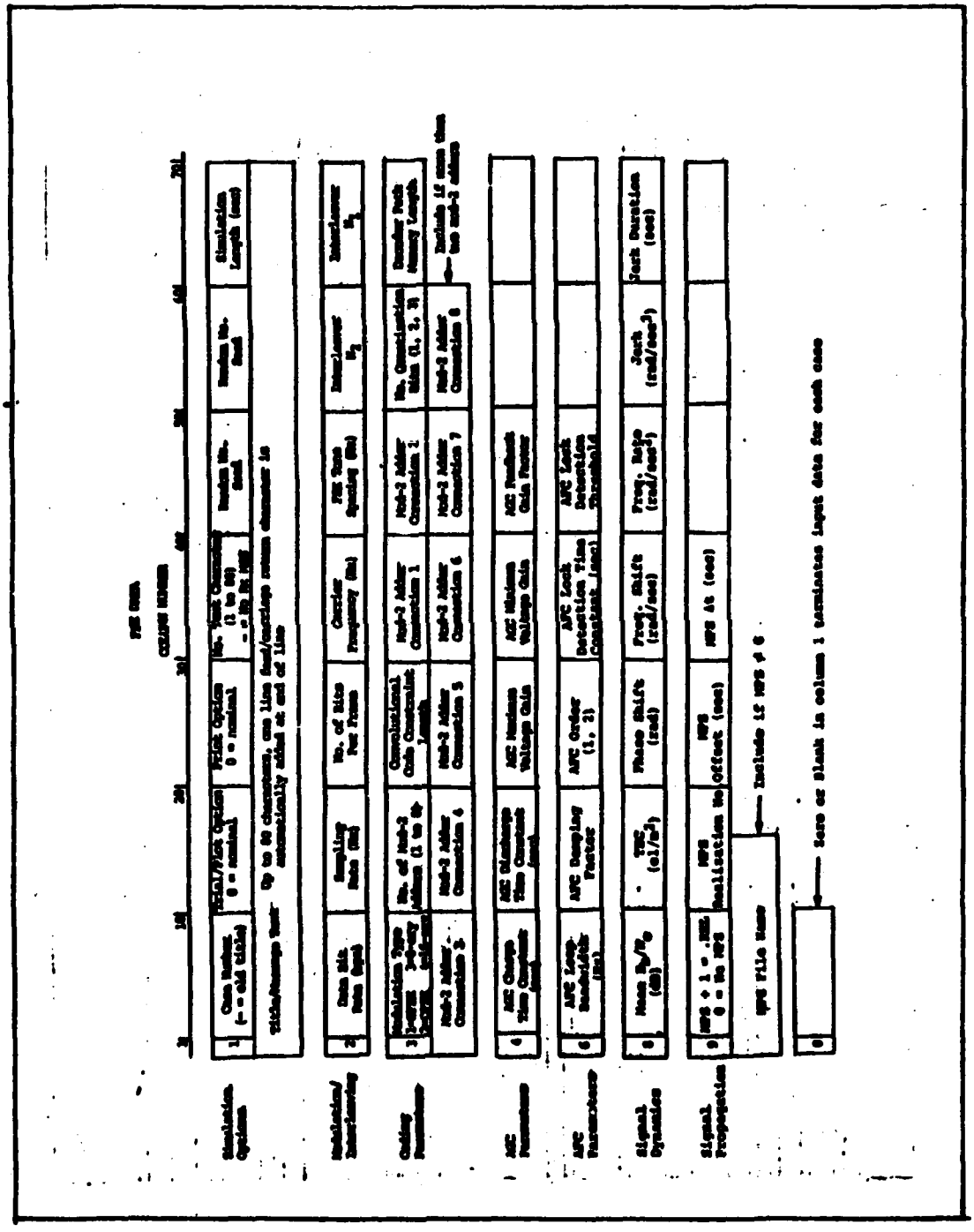
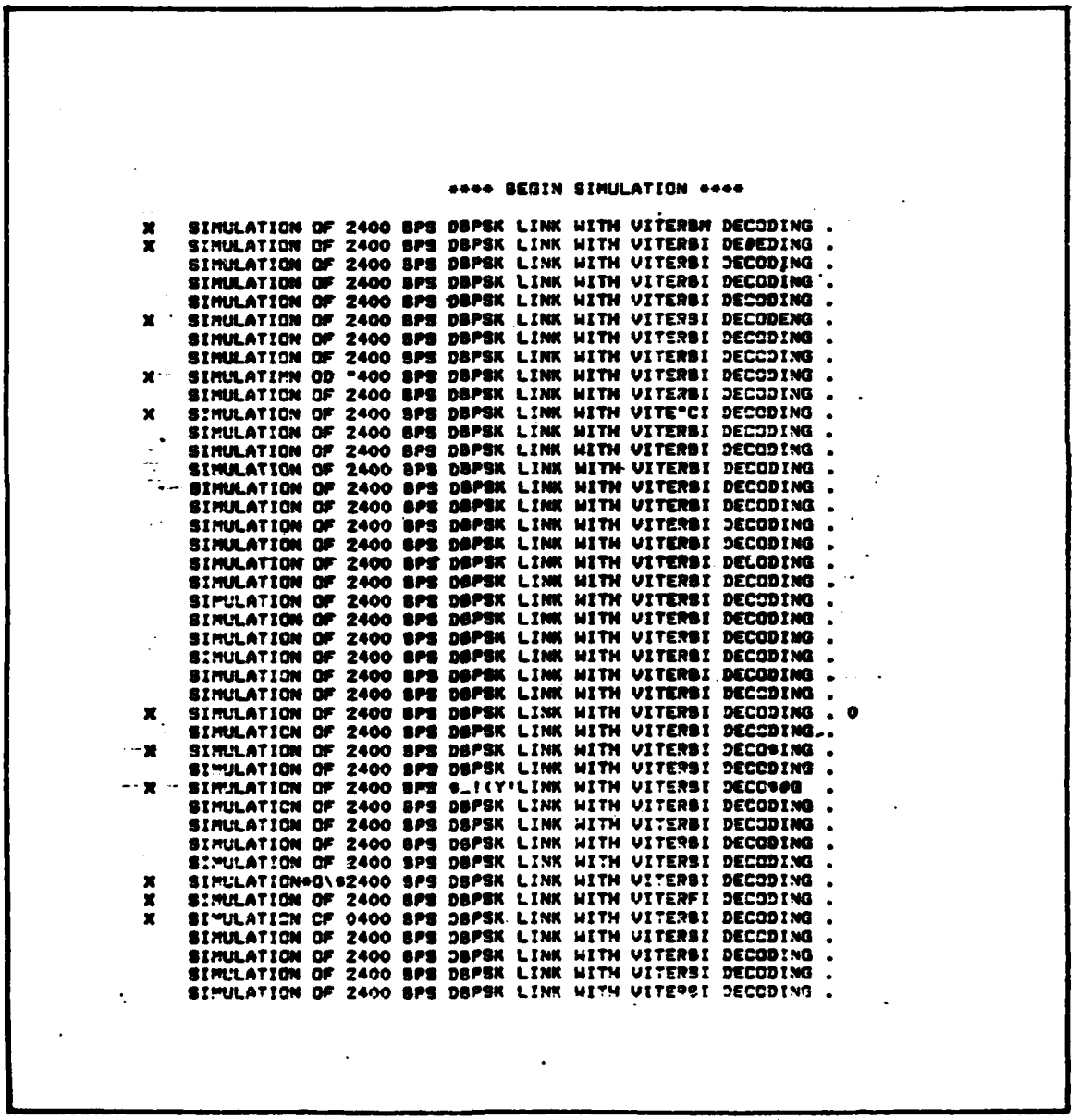
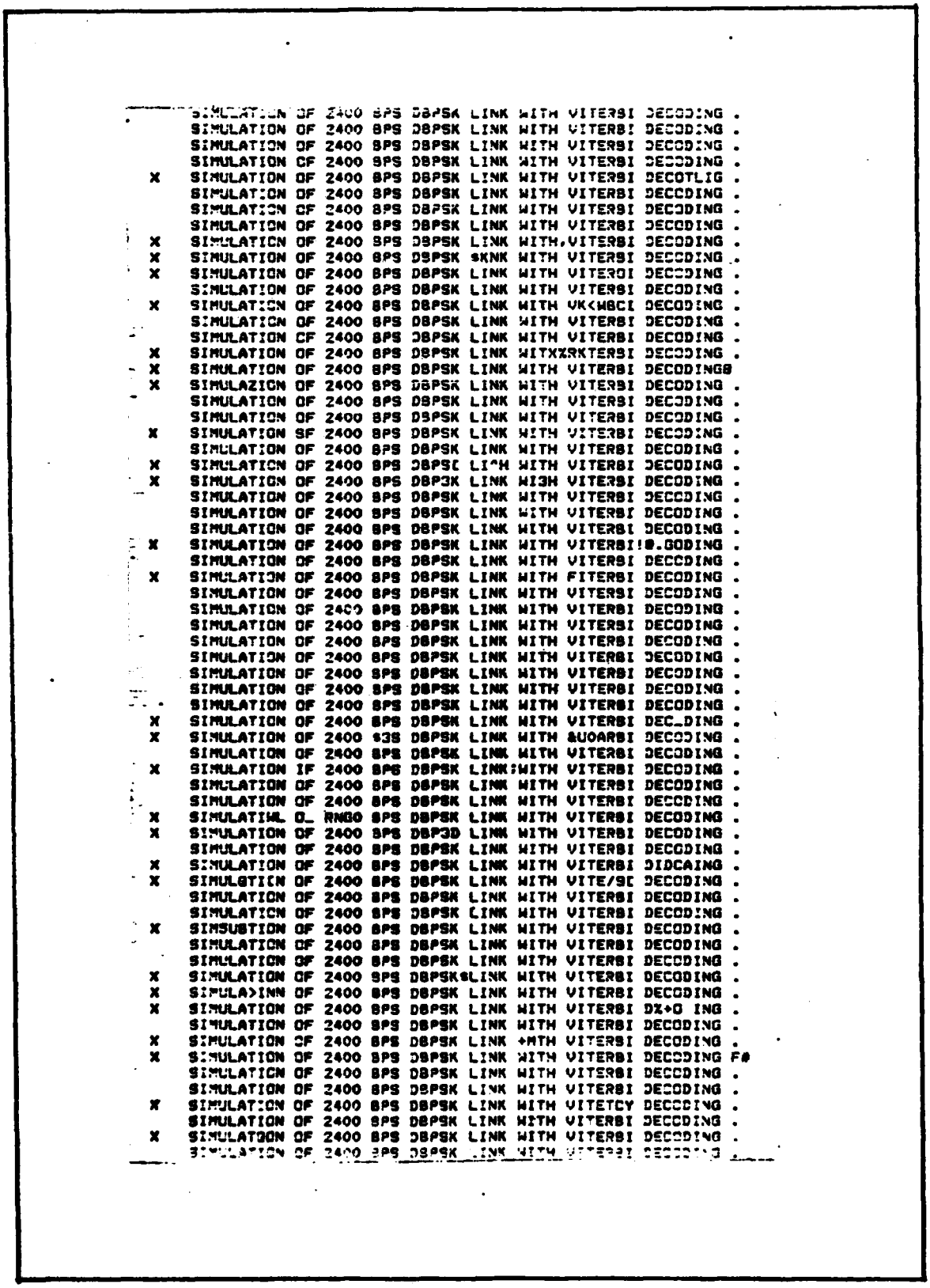


Figure A.2. FSK Data File

Appendix B. Sample of Simulation Output for a Scintillated Channel, $\tau_o = 0.016$ sec.

```
PSK NO.      827   27-AUG-82   17:11:43     SIMULATION LENGTH (SEC)  2.25379E+01

SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .

                          RECEIVER DESIGN PARAMETERS
TYPE OF PSK MODULATION          DBPSK       DIFFERENTIAL  ENCODING          DPSK
PSK DATA BIT RATE (BPS)  2.40000E+03        CONVOLUTIONAL ENCODING           YES
SYMBOL   BIT RATE (BPS)  4.80000E+03        NUMBER DATA BITS/FRAME           880
A/D SAMPLING RATE  (HZ)  9.60000E+03        CARRIER FREQUENCY (HZ)   2.00000E+10

CODE CONSTRAINT LENGTH              7       NUMBER OF MOD-2 ADDERS             2
NO. SOFT-DECISION BITS              3       OCTAL CONNECT PATTERNS       133 171

INTERLEAVER PARAMETERS  TYPE4(133,61)       DECODER PATH DATA BITS            31

AGC TIME CONSTANT (SEC)  1.00000E+01        DBPSK LOCK DETECT (SEC)  0.00000E+00


  PLL LOOP BANDWIDTH (HZ)  0.00000E+00        AFC LOOP BANDWIDTH (HZ)  3.00000E+00
  PLL LOOP DAMPING FACTOR  0.00000E+00        AFC LOOP DAMPING FACTOR  1.00000E+00
  PLL LOOP ORDER                  0           AFC LOOP ORDER                    2
  PLL LOCK DETECTION(SEC)  0.00000E+00        AFC LOCK DETECTION(SEC)  1.00000E+00
  PLL LOCK THRESHOLD       0.00000E+00        AFC LOCK THRESHOLD       1.50000E-01
  PLL LOOP CONFIGURATION          OFF         AFC SWITCH POSITION              ON

                        RECEIVED SIGNAL PARAMETERS
  MEAN VALUE OF EB/NO(DB)  8.00000E+00        MPS FILE  BOS:PCM.REL
  INTEGRATED TEC  (EL/M2)  0.00000E+00        MPS IDENTIFICATION NO.         8091
  PHASE SHIFT    (RADIANS) 0.00000E+00        MPS REALIZATION NUMBER            1
  FREQUENCY SHIFT (RAD/S)  0.00000E+00        MPS TIME OFFSET   (SEC)  0.00000E+00
  FREQUENCY RATE (RAD/S2)  0.00000E+00        MPS TIME INTERVAL(SEC)   2.58000E-04
  FREQUENCY JERK (RAD/S3)  0.00000E+00        MPS GRID INTERVAL   (M)  1.83105E+00
  JERK DURATION     (SEC)  0.00000E+00        MPS MAXIMUM TIME (SEC)   4.22681E+00
  NO. OF DFT FREQUENCIES              0       NO. OF MPS FREQUENCIES            1
  FREQUENCY SPACING  (HZ)  0.00000E+00        MPS INPUT TYPE OPTION             1

  RANDOM NUMBER SEED     57815    28088       OUTPUT OPTIONS           0        0
```


**** BEGIN SIMULATION ****
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .


SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECOTLIG .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH,VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK $KNK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VK<WBCI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITXXRKTERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING@
X SIMULAZION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION SF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSC LI^H WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBP3K LINK WI3H VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI!@.GODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH FITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DEC_DING .
X SIMULATION OF 2400 *3S DBPSK LINK WITH &UOARBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION IF 2400 BP6 DBPSK LINK:WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATIWL O_ RNGO BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBP3D LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DIDCAING .
X SIMULGTICN OF 2400 BPS DBPSK LINK WITH VITE/9C DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMSUBTION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 SPS DBPSK$LINK WITH VITERBI DECODING .
X SIMULA>INN OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI D%+O ING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK +MTH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING F@
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITETCY DECODING .
SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .

```
A   SIMULATION OF 2400PVYS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X   SIM-1:NMCN OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X   SIMULATICN OF 2400 BPS DBPSK LINK W)W"'VITERBI DECODING .
X   SIMULATION OF 2400 BPS TGPSK L)NK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING /
X   SIMULATION OF 2400 BPS DBPSKO<KNK WITH VITERBI DECODING .
X   SIMU$DIION OF 3400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X   SIMULATION CF 2400 BPS DBPSK@JHMK WITH VITERBI DECODING .
X   SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING . H!
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
X   SIMULATION CF 2400 BPS DBPSK LINK WITH VITERBI DECODI>C".
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION CF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION CF 2400 BPS DBPSK LINK WITH VITERBI DECODING .
    SIMULATION OF 2400 BPS DBPSK LINK WITH VITER

DEMODULATOR ERROR SUMMARY   SYMBOLS     BITS       CHARS      LINES      FRAMES
    TOTAL NO. RECEIVED      108182     50040       9340        136         52
    TOTAL NO. IN ERROR       12775       263        131         49         36
    ERROR RATE OBSERVED    1.181E-01  5.258E-03  1.571E-02  3.603E-01  6.923E-01

STATISTICAL ERROR RATES       MEAN        MIN        MAX
    UNCODED DPSK           1.187E-01  3.584E-11  5.000E-01

TRACKING ERROR STATISTICS     MEAN        RMS        SIGMA      MAXIMUM
    PHASE (RADIANS)        -6.929E+02  7.335E+02  2.407E+02  1.187E+03
    FREQUENCY (HZ)         -4.315E+00  6.174E+01  6.158E+01  2.031E+03

MEASURED SIGNAL STATISTICS    MEAN        RMS      SIGMA/S4     MINIMUM     MAXIMUM
    PHASE (RADIANS)        4.404E+01  5.368E+01  3.065E+01  -2.289E+01  1.027E+02
    FREQUENCY (HZ)         8.737E-01  4.429E+01  4.429E+01  -1.488E+03  1.912E+03
    EB/NO AT  INPUT        7.992E+00 DB          1.004E+00
    AGC LEVEL             -3.687E+00 DB                     NO. SAMPLES     216364

SEED VALUES AT END OF RUN      31392      33518            ELAPSED TIME 3755.594
```

Vita

James A. Frazier, Jr. was born on 8 March 1955 in Gettysburg, Pennsylvania. He graduated from Biglerville High School in 1973 and then attended The Pennsylvania State University, from which he received the degree of Bachelor of Science in Electrical Engineering in November 1977. He was commissioned in the U.S. Air Force in November 1977 and subsequently completed the Communications-Electronics Officer Engineering Course at Keesler AFB, Mississippi in June 1978. Captain Frazier was then assigned to the Air Force Weapons Laboratory, Kirtland AFB, New Mexico, where he worked in satellite communications link survivability/vulnerability in a nuclear environment. In June 1981 he entered the School of Engineering, Air Force Institute of Technology. Captain Frazier has been selected for an assignment in the Satellite Communications Systems Development Division at the Defense Communications Engineering Center in Reston, Virginia. He is married to Kathy Jean and they have a daughter, Jeanine Lynn.

Permanent Address: R.D. #1 Box 80
Aspers, PA 17304

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GE/EE/82D-32	2. GOVT ACCESSION NO. A124814
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Performance of Sequentially Decoded Long Constraint Length Codes for a Satellite Communications Link in a Scintillated Channel	5. TYPE OF REPORT & PERIOD COVERED: MS THESIS
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James A. Frazier, Jr. Captain USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Weapons Laboratory Satellite and C^3 Branch (AFWL/NTYCC) Kirtland AFB, NM 87117	12. REPORT DATE: Dec 1982
	13. NUMBER OF PAGES: 218
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release IAW AFR 190-17.

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

4 JAN

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Convolutional Coding	Signal Scintillation	Synchronous Interleaving
Sequential Decoding	Fading Channel	
Viterbi Decoding	Interleaving	EHF Satellite Communications
Link Simulation	Phase-Shift-Keying	
Satellite Communication Links		Fano Algorithm

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Convolutional codes with long constraint lengths, on the order of 25 to 30, can be decoded using the Fano sequential decoding algorithm. While some increase in coding gain over short codes may be achieved for small probabilities of bit error, sequential decoding is not optimum and does not perform as well as the Viterbi decoding algorithm. However, because of power limitations on satellite downlinks, the possibility of achieving some additional gain with long constraint length codes for the scintillated

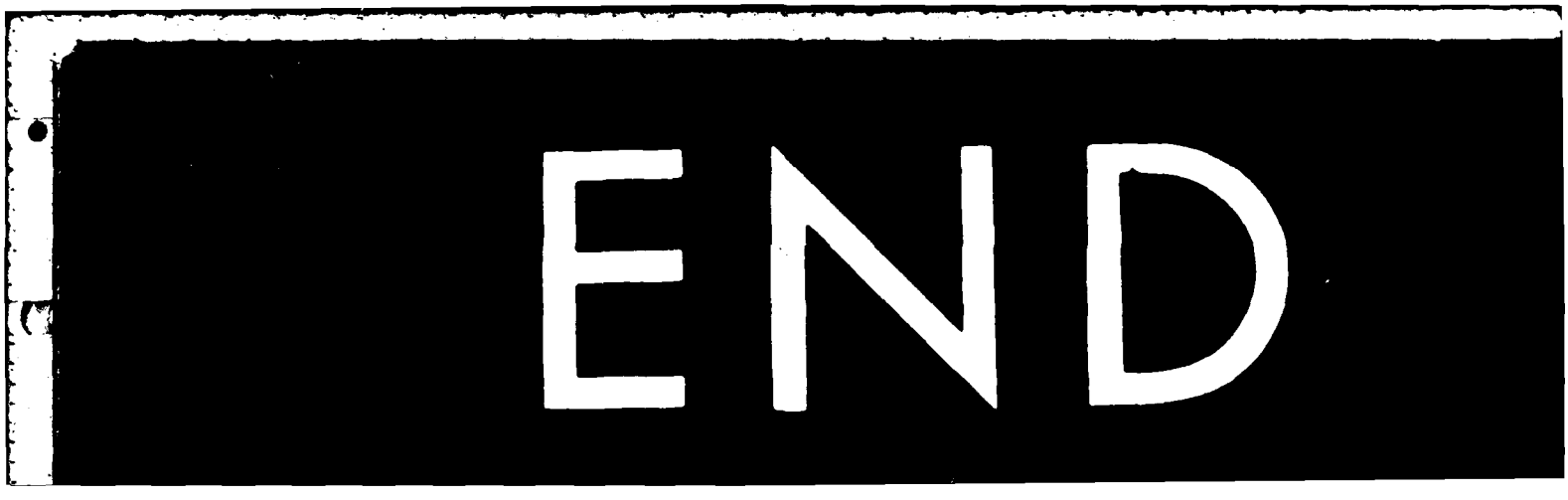
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

signal channel is receiving increasing interest.

This report presents an analysis of the performance and implementation trade-offs between short codes with Viterbi decoding and long codes with sequential decoding in an additive white Gaussian noise channel and a scintillation channel created by a high-altitude nuclear detonation. Performance is provided for an EHF, 2400 bps, DBPSK modulated satellite communications link in terms of bit error rates and decoded bit processing delays.

Performance comparisons between the two convolutionally encoded systems show Viterbi decoding outperforms the sequentially decoded link scintillated signal fading channels at bit error probabilities greater than 10^{-5}. The Viterbi decoded link, however, loses its advantage over the sequentially decoded systems at bit error rates less than 10^{-5} in slow fading channel conditions.


END